ioProgrammo Anno IX - N°6 (92) Mobile Camera • VideoGames con il Vertex Shader • System hacking con gli hook di sistema • Intel C++ Compiler • Vocal Visual Basic EDIZIONI MASTER

PER ESPERTI E PRINCIPIANTI

Poste Italiane • Spedizione in a.p. - 45% • art. 2 comma 20/b legge 662/96 - AUT. N. DCDC/033/01/CS/CAL Periodicità mensile • GIUGNO 2005 • ANNO IX, N.6 (92)

MOBILE CAMERA

IMPARA A PROGRAMMARE LA CAM DEL CELLULARE E CREA UN BLOG FOTOGRAFICO SU INTERNET

- Gestisci la fotocamera con una tua applicazione
- Salva le immagini e i video all'interno del telefono
- Crea un servizio sul web pronto a ricevere le foto

HACKING DI SISTEMA

Prendi il controllo di Windows intercettando gli eventi e riprogrammandoli con un hook

VIDEOGAMING

Sfrutta al massimo l'hardware dei PC e crea giochi superveloci con Vertex Shader e DirectX

VISUAL BASIC

FAI PARLARE IL PC

Ecco come creare software moderno capace di supportare gli utenti tramite la voce

C++

INTEL C++ COMPILER

La guida per utilizzare al meglio la potenza di calcolo di un compilatore ottimizzato

IOPROGRAMMO WEB

QUANDO PHP INCONTRA JAVA

Impara come riutilizzare il tuo lavoro inserendo classi Java nelle pagine PHP

C# & VB.NET

EXCEL & C# COPPIA VINCENTE

Usa tutta la potenza di .NET per rielaborare i dati di un foglio elettronico

NON SIAMO SOLI: REMOTING .NET

Metti in comunicazione i software e crea applicazioni distribuite

TUTTI PRONTI PER MONO

Crea programmi in C#, che funzionano sia con Linux sia con Windows

JAVA

GESTIRE LA POSTA ELETTRONICA

Sfrutta JavaMail per spedire e ricevere la posta direttamente da Java

INTERFACCE : CON XML SI PUÒ!!!

Alla scoperta di Thinlet, una libreria per definire il layout usando XML

ACCESSO AI DB FACILE CON JDBC

Elimina ogni problema sui database, imparando come usarli semplicemente

I CORSI PER IMPARARE
- VISUAL BASIC.NET 2003
- SYMBIAN
- ASP.NET

KERNEL: COME RISOLVERE I PROBLEMI DEL MULTITASKING

Uno sguardo avanti

XML, soap, web services, programmazione internet oriented, applicazioni dedicate ad ogni tipo di mercato e soluzioni sperimentate per ogni tipo di device, negli ultimi anni le abbiamo provate tutte per far uscire il mercato da una stasi che tende a diventare cronica. Pur scontrandoci con le diffidenze di amministratori un po' troppo tradizionalisti, pur facendo i conti con clienti ancorati a vecchi modi di concepire la gestione aziendale, noi programmatori siamo riusciti in un modo o nell'altro ad evolvere e a fare evolvere questa vecchia Europa nella direzione del progresso. E non credo di essere presuntuoso nel dire che chi, come noi, come voi, fa ricerca e sviluppo non sia una parte importante del propulsore che spinge verso una qualità della vita migliore. E adesso? adesso che in campo software quasi tutto è stato inventato, riusciremo ancora a trovare nuovi canali dove spingere la nostra curiosità? Come sempre, un nuovo settore, sembra nascere da un evento esclusivamente commerciale: l'acquisizione di Macromedia da parte di Adobe. Questa acquisizione vuol dire una sola cosa, ovvero che il mondo della grafica può considerarsi saldamente nelle mani di un grande colosso. Questo per noi vuol dire standard, vuol dire per noi possibilità di sviluppare plugin, applicazioni, estensioni, formati, dedicati al mondo della grafica. Che sia questa una delle tante nuove frontiere che ci attendono?

Fabio Farnesi

All'inizio di ogni articolo, troverete un simbolo che indicherà la presenza di codice e/o software allegato, che saranno presenti sia sul CD (nella posizione di sempre *\soft\codice* e *\soft\tools*) sia sul Web, all'indirizzo http://cdrom.ioprogrammo.it.
Per scaricare software e codice da Internet, ogni mese indicheremo una password differente. Per il numero che avete fra le mani la combinazione è:

Username: **micio74** *Password:* **rumors82**

Impara a programmare la cam del tuo cellulare e crea un blog fotografico sul web

- **Prendi il controllo della fotocamera**
- **Salva le foto e il video sul cellulare**
- **Crea un servizio web pronto a ricevere le foto**

pag. 14

Anno IX - N.ro 6 (92) - Giugno 2005 - Periodicità: Mensile
Reg. Trib. di CS al n.ro 593 del 11 Febbraio 1997
Cod. ISSN 1128-594X
E-mail: ioprogrammo@edmaster.it
http://www.edmaster.it/ioprogrammo
http://www.ioprogrammo.it

Direttore Editoriale: Massimo Sesti
Direttore Responsabile: Massimo Sesti
Responsabile Editoriale: Gianmarco Bruni
Redazione: Raffaele del Monaco, Fabio Farnesi
Collaboratori: R. Allegra,D. Berta, M. Bigatti, D. Bochicchio, L. Buono, D. De Michelis, F. C. Ferracchiati, F. Grimaldi, A. Lacava, E. Lancione, M. Locuratolo, A. Marroccelli, V. Muraglia, F. Paparoni, A. Pelleriti, P. Perrotta, A. Trapani, F. Vaccaro, S. Vena, V. Vessia, V. Viola.
Segreteria di Redazione Veronica Longo

Realizzazione grafica: Cromatika S.r.l.
Responsabile grafico: Paolo Cristiano
Coordinamento tecnico: Giancarlo Sicilia
Illustrazioni: M. Veltri
Impaginazione elettronica: Aurelio Monaco

"Rispettare l'uomo e l'ambiente in cui esso vive e lavora è una parte di tutto ciò che facciamo e di ogni decisione che prendiamo per assicurare che le nostre operazioni siano basate sul continuo miglioramento delle performance ambientali e sulla prevenzione dell'inquinamento"

Certificato UNI EN ISO 14001
N. 9191 CRMT

Realizzazione Multimediale: SET S.r.l.
Coordinamento Tecnico: Piero Mannelli
Realizzazione CD-Rom: Paolo Iacona

Pubblicità: Master Advertising s.r.l.
Via Ariberto, 24 - 20123 Milano
Tel. 02 831212 - Fax 02 83121207
e-mail advertising@edmaster.it
Sales Director: Max Scortegagna
Segreteria Ufficio Vendite: Daisy Zonato

Editore: Edizioni Master S.p.a.
Sede di Milano: Via Ariberto, 24 - 20123 Milano
Tel. 02 831212 - Fax 02 83121206
Sede di Rende: C.da Lecco, zona industriale - 87036 Rende (CS)
Presidente e Amministratore Delegato: Massimo Sesti

ABBONAMENTO E ARRETRATI
ITALIA: Abbonamento Annuale: ioProgrammo (11 numeri) €39,90 sconto 26% sul prezzo di copertina di €53,90
Offerte valide fino al 30/06/05
Costo arretrati (a copia): il doppio del prezzo di copertina + € 5.32 spese (spedizione con corriere). Prima di inviare i pagamenti, verificare la disponibilità delle copie arretrate allo 02 831212.
La richiesta contenente i Vs. dati anagrafici e il nome della rivista, dovrà essere inviata via fax allo 02 83121206, oppure via posta a EDIZIONI MASTER via Ariberto, 24 - 20123 Milano, dopo avere effettuato il pagamento, secondo le modalità di seguito elencate:

- cc/p n.16821878 o vaglia postale (inviando copia della ricevuta del versamento insieme alla richiesta);
- assegno bancario non trasferibile (da inviarsi in busta chiusa insieme alla richiesta);
- carta di credito, circuito VISA, CARTASI', MASTERCARD/EUROCARD, (inviando la Vs. autorizzazione, il numero della carta, la data di scadenza e la Vs. sottoscrizione insieme alla richiesta).
- bonifico bancario intestato a Edizioni Master S.p.A. c/o Banca Credem S.p.a. c/c 01 000 000 5000 ABI 03032 CAB 80880 CIN Q (inviando copia della distinta insieme alla richiesta).

SI PREGA DI UTILIZZARE IL MODULO RICHIESTA ABBONAMENTO POSTO NELLE PAGINE INTERNE DELLA RIVISTA. L'abbonamento verrà attivato sul primo numero utile, successivo alla data della richiesta.

Sostituzioni: qualora nei prodotti fossero rinvenuti difetti o imperfezioni che ne limitassero la fruizione da parte dell'utente, è prevista la sostituzione gratuita, previo invio del materiale difettato.
La sostituzione sarà effettuata se il problema sarà riscontrato e segnalato entro e non oltre 10 giorni dalla data effettiva di acquisto in edicola e nei punti vendita autorizzati, facendo fede il timbro postale di restituzione del materiale.
Inviare il CD-Rom difettoso in busta chiusa a:
Edizioni Master - Servizio Clienti - Via Ariberto, 24 - 20123 Milano

Assistenza tecnica: ioprogrammo@edmaster.it

Servizio Abbonati:
tel. 02 831212
@ e-mail: servizioabbonati@edmaster.it

Stampa: Rotoeffe Via Variante di Cancelliera, 2/6 - Ariccia (Roma)
Stampa CD-Rom: Neotek S.r.l.- C.da Imperatore - zona ASI Bisignano (CS)
Distributore esclusivo per l'Italia: Parrini & C S.p.A.
Via Vitorchiano, 81 - Roma

Finito di stampare nel mese di Maggio 2005

Edizioni Master edita: Computer Bild, Idea Web, Go!OnLine Internet Magazine, Win Magazine, Quale Computer, DVD Magazine, Office Magazine, La mia Barca, ioProgrammo, Linux Magazine, Softline Software World, HC Guida all'Home Cinema, MPC, Discovery DVD, Computer Games Gold, inDVD, I Fantastici CD-Rom, PC VideoGuide, I Corsi di Win Magazine, I Filmissimi in DVD, Filmteca in DVD, La mia videoteca, TV e Satellite, Win Extra, Home entertainment, Digital Japan, Digital Music, Horror Mania, ioProgrammo Extra, Le Collection.

VIDEOGAMING

Sfrutta al massimo l'hardware del PC con il vertex shader e le directx

ioProgrammo cerca articolisti freelance competenti nei seguenti argomenti:
Javascript, Python, Perl, ASP.NET, PHP, Flash, Security
Inviare curriculum dettagliato a
ioProgrammo@edmaster.it

RUBRICHE

QUALCHE CONSIGLIO UTILE

I nostri articoli si sforzano di essere comprensibili a tutti coloro che ci seguono. Nel caso in cui abbiate difficoltà nel comprendere esattamente il senso di una spiegazione tecnica, è utile aprire il codice allegato all'articolo e seguire passo passo quanto viene spiegato tenendo d'occhio l'intero progetto. Spesso per questioni di spazio non possiamo inserire il codice nella sua interezza nel corpo dell'articolo. Ci limitiamo a inserire le parti necessarie alla stretta comprensione della tecnica.

RIVISTA + CD-ROM in edicola

IL SOFTWARE DEL MESE

Questo mese ioProgrammo vi regala GFI: Network Server Monitor 6 con licenza completa per tre server. Un omaggio del valore di 215 €

GFI Network Server Monitor 6 è un software per il monitoring del corretto funzionamento dei server che forniscono servizi alla rete aziendale. È possibile controllare sia sistemi Windows che Linux e praticamente l'intera gamma dei servizi disponibili, dal Web Server al Mail Server, file Server e tutti gli altri servizi tipici di unambiente di rete.

I contenuti del CD-Rom

Apache 1.3.33/2.0.53
Il Web Server più usato
Directory: /Apache

Python 2.4.1.
Il linguaggio emergente
Directory: /Python

Ultimate ++
L'IDE più innovativo per C++
Directory: /Ultimatepp

Lazarus 0.9.6
Il clone FreeWare di Delphi
Directory: /lazarus

Tomcat 5.5.9
L'application Server per JSP
Directory: Directory /Tomcat

Eclipse 3.0.2
Il più innovativo del momento
Directory: Directory /Eclipse

DevCPP 4.9.9.2
l'ambiente più usato in C++
Directory: Directory: /DevCPP

PHP 5.0.4
Il linguaggio per il Web
Directory: Directory /PHP

SharpDevelop 1.0.3
L'alternativo per C#
Directory /SharpDevelop

J2SE 1.5.0.2
Indispensabile per Java
Directory /J2SE

DevPHP
Leggerissimo per PHP
Directory /DevPHP

Dadabik 3.2
Un creatore di interfacce
Directory: /DadaBik

Mysql 4.1.11 – Beta 5.0.3
Il DB server di riferimento
Directory: MySQL

Firebird 1.5
Un db multipiattaforma
Directory: /Firebird

Hibernate 3.0
Per la persistenza dei dati
Directory: /Hibernate

Jasper Reports 0.6
Per creare reports in Java
Directory /Jasperreports

Snippet Compiler 2
Compila un pezzo di codice
Directory: /SnippetCompiler

Notepad++ 2.9
Editor per sviluppatori
Directory: /Notepad

Ndoc
Per creare codice documentato
Directory: /Ndoc

Nant
L'assemblatore di progetti
Directory /nant

Nunit
Il costruttore di test
Directory /nunit

SysLinux 3.0.7
Installare sotto linux
Directory /SysLinux

Regulator 2.0.3
No problem con le regular expression
Directory /Regulator

Eclipse Visual Editor
Per rendere Eclipse RAD
Directory /Visual Editor

SuperEdi 3.7
Un editor piccolo e funzionale
Directory /Superedi

Spe 0.7.3
L'editor per Python
Directory /SpeEditor

wxWidgets 2.4.2
Per lo sviluppo di interfacce
Directory: /WxWidgets

EasyPHP 1.8
Per installare un server PHP
Directory /EasyPHP

Mono 1.6.0
.NET su tutte le piattaformeforme
Directory: /Mono

PHPEclipse 1.1.3
Per usare Eclipse con PHP
Directory /PHPEclipse

WinMerge 2 2.2
Mai più versioni di codice differenti
Directory /WinMerge

J2ME Polish
Il costruttore di GUI per device mobile
Directory /J2MEPolish

Installer2Go
Un sistema per realizzare pacchetti di installazione
Directory /Installer2Go

Jedit 4.2
L'editor leggero per programmatori Java
Directory /Jedit

XAMPP for Windows 1.4.13
ServerWeb, FTP, DBMS, PHP e Perl tutto in uno
Directory: /Xampp

EMF
Eclipse Modeling Framework
Directory: EMF

PHP Java Bridge
Il ponte fra Java e PHP
Directory /PHPJavaBridge

PHPGroupware 0.9.16
Directory /PHPGroupware

PHTML Encoder 3.8
Per criptare le vostre pagine PHP
Directory /PHTMLEncoder

Mrtg 2.11
Per tenere sotto controllo il consumo delle risorse
mrtg

Mantis 0.19
Ottima soluzione per la gestione dei Bug
Mantis

MediaWiki 1.4.0
Per la creazione di documentazione
Directory /MediaWiki

James 2.2.0
Il server di posta di Apache
Directory /James

Icecast 2.2.0
Il server di streaming
Directory: Icecast

SquirrelMail
La capostipite delle WebMail
Directory /SquirrelMail

Struts 1.2.4
Il FrameWork per costruire applicazioni conformi all'MVC
Directory /Struts

PircBot 1.4.4
Un bot irc programmabile in Java
Directory /PircBot

Drupal 4.6.0
Il principe delle piattaforme di Blogging
Directory: /Drupal

Thinlet
Le API per creare interfacce usando XML
Directory /Thinlet

Dia 0.94
Crea diagrammi di flusso con semplicità
Directory: Dia

Gef
Graphical Editing Framework
Directory /Gef

Irrlicht 0.9
La nuova versione del motore 3D più in voga del momento
Directory /irrlicht

News

MICROSOFT RILASCIA IRONPYTHON

Anche questa è una news che scotta. Che Python fosse un linguaggio emergente, non c'erano dubbi. Che fosse talmente emergente e talmente ben considerato che persino Microsoft rilasciasse un'implementazione del linguaggio da far girare su piattaforma .NET questo era molto meno probabile, ed invece all'indirizzo *http://workspaces.gotdotnet .com/ironpython* è possibile proprio scaricare una pre-alpha release del linguaggio da integrare nel .NET framework. Potrebbe essere una mossa azzeccata per portare una parte degli sviluppatori OpenSource, che sono notoriamente l'avversario più temibile di MS a sviluppare applicazioni utilizzando strumenti nativi per l'OS di casa Redmond. Come si suol dire, se non li puoi abbattere, fatteli amici.

ARRIVA ENDIAN FIREWALL

L'annuncio dell'arrivo di Endian Firewall ha suscitato un certo scalpore nella comunità informatica. Il motivo principe dell'interesse suscitato da questo Firewall nasce dal fatto che è il primo sistema di sicurezza italiano opensource e con licenza GPL ad essere stato pensato per girare su sistemi Embedded.
Endian si è concentrata nell'evoluzione di IPCop, già nota per essere una distribuzione Linux OpenSource dedicata alla sicurezza. Le migliorie apportate da Endian hanno portato alla creazione di un sistema composto in 160 pacchetti e concentrato in 100MB interamente dedicato alla security, ideale per girare su sistemi Embedded come su macchine ormai datate

RILASCIATO IL GCC 4.0

**Il GCC è il cuore pulsante di ogni progetto OpenSource che si rispetti. Non solo, spesso e volentieri soluzioni anche ideate per i sistemi Windows vengono costruite utilizzando porting del GCC.
Fatte queste premesse si intuisce il clamore che il rilascio di una major version suscita nel mondo della programmazione. Il CHANGELOG ovvero le note che accompagnano il rilascio e che descrivono i cambiamenti rispetto alla precedente versione è di dimensioni generose ed è reperibile all'indirizzo http://gcc.gnu.org/gcc-4.0/changes.html. Tuttavia fra le novità è importante sottolineare il supporto a SSA, Static Single Assignment Form. SSA è un argomento piuttosto complesso, riassumiamo brevemente che questa innovazione ottimizza la generazione del codice, aumentando la velocità dell'applicativo.**

CENTO MILIONI DI DOWNLOAD PER SKYPE

Che il Voice Over IP fosse uno degli elementi chiave per il rinnovo dei servizi all'utenza comune era palese. In Italia diverse compagnie avevano tentato di lanciare servizi basati su VOIP, nessuna aveva ottenuto il successo che sta ottenendo Skype. Peccato che in Italia forse i tempi non siano ancora sufficientemente maturi per apprezzare il servizio. In Danimarca, Finlandia, Francia, Hong Kong, Norvegia, Svezia, Regno Unito e Stati Uniti gli utenti possono già richiedere fino a tre numeri di telefono per essere sempre raggiungibili anche da utenti non appartenenti al circuito di skype.

ADOBE ACQUISTA MACROMEDIA

Adobe Systems Incorporated leader nel campo della grafica professionale per la carta stampata soprattuto con software del tipo Adobe Photoshop, Adobe Illustrator, Adobe Indesign ha annunciato di avere raggiunto un accordo definitivo per l'acquisizione di Macromedia, leader indiscusso nel campo della grafica per il Web con software del calibro di Macromedia Flash, Macromedia DreamWeaver, Macromedia Fireworks. Il valore della transazione si aggira intorno ai 3.4 miliardi di dollari. Non ci sarà però uno scambio fisico di denaro, di fatto l'intera operazione si concluderà scambiando azioni. Sono previste 0,69 azioni di Adobe per ogni azione di Macromedia, con la valutazione di un titolo Macromedia in 41,86$. Rimane da decidere la composizione del consiglio d'amministrazione, ma i bene informati sostengono che la direzione rimarrà saldamente nelle mani di Adobe. Tuttavia i rappresentanti di Macromedia godranno di una presenza significativa. Al di la del valore strettamente economico dell'operazione è importante valutare il senso com-

AGGIORNAMENTI PER WINDOWS UPDATE

Microsoft ha appena rilasciato la prima Release Candidate del nuovo Windows Server Update Services, nome in codice WSUS. Si tratta del servizio che consente agli utenti Windows di mantenere aggiornato il proprio computer. Come noto il servizio si basa su due componenti, uno installato lato client e un lato web.
Attraverso uno scambio di dati, il client scarica dal server sempre le versioni aggiornate dei componenti di sistema, questo consente di tenere la propria macchina sempre aggiornata con le ultime patch di sicurezza e con le release più recenti dei prodotti. Nella nuova versione è previsto un aumento della larghezza di banda e quindi una conseguente maggiore velocità nello scaricare gli aggiornamenti, un supporto migliorato verso il multilingua e un range di prodotti allargati.
Chi volesse provare il nuovo servizio può registrarsi all'indirizzo *http://www.microsoft .com/windowsserversystem /updateservices/evaluation/trial/default.mspx*.

merciale di questa acquisizione. L'unione delle due compagnie configura un colosso capace di contrastare in modo inequivocabile le mosse di chiunque altro volesse tentare la scalata al campo della grafica, fosse essa rivolta al Web, alla carta stampata o alla televisione.

Per quello che riguarda l'utenza comune, questa fusione non può che tracciare una linea verso applicazioni sempre più interessanti. Adobe e Macromedia rappresentano da sole la massima espressione della creatività nel campo degli strumenti dedicati alla grafica, e la fusione di elementi così geniali non potrà che portare alla creazione di software sempre più utile e sofisticato.

D'altro canto registriamo una clamorosa assenza di concorrenza che ci incute, come sempre in questi casi, un moderato timore.

OTTAVA VERSIONE PER OPERA

Nella guerra fra browser fra un Firefox che avanza, un Internet explorer che annuncia la sua versione numero 7, non poteva mancare il rilascio di Opera release numero 8. Forse un po troppo tardi aggiungiamo noi. Firefox e Internet Explorer non sono concorrenti a cui si possa concedere un minimo vantaggio. E comunque le novità di questa nuova versione di Opera sono rilevanti. Prima di tutto si parla di una funzionalità per il resizing automatico delle pagine, ovvero riducendo la finestra del browser si avrà contestualmente un ridimensionamento degli oggetti che la compongono. Gli altri miglioramenti coinvolgono la sicurezza. Opera dichiara che il browser è stato progettato con un occhio particolare alla sicurezza degli utenti che lo utilizzano. Ancora in evidenza la possibilità di comandare il browser utilizzando la voce. Infine Opera rimane ovviamente multipiattaforma, funziona sia su Windows che su Linux che su Mac OS. Sicuramente si tratta di un browser tecnologicamente evoluto, il primo ad avere portato alla ribalta le tabsheet per la navigazione, e anche questa nuova versione non tradisce le aspettative. Certo adesso riconquistare le posizioni perse a favore di Explorer e soprattutto di Firefox non sarà cosa da poco.

NASCE IL VISUAL BASIC 6.0 RESOURCE CENTER

La volontà di Microsoft di volere favorire il passaggio degli sviluppatori Visual Basic al mondo .NE T è evidente. L'annuncio della creazione di questo Visual Basic Resource Center che fisicamente risponde all'indirizzo http://msdn.microsoft.com/VBRun/ segue quello che aveva esplicitato la volontà del colosso di Redmond di cessare il supporto a VB6. La creazione di questo polo non indica un cambio di rotta nelle politiche di Microsoft, semmai mostra come la procedura di passaggio da VB6 alla piattaforma .NET stia procedendo talmente a rilento da costringere la società di Bill Gates a creare strumenti che gradualmente mettano in grado i programmatori di abbracciare la nuova tecnologia. Se da un lato nel VBRun trovano posto esempi di applicazioni “notevoli” scritte in VB6, dall'altro un'intera sezione è dedicata all'uso congiunto di tecnologie .NET con quelle classiche. Il senso è evidente, mostrare agli sviluppatori VB6 come e quanto la nuova tecnologia .NET possa essere utilizzata per migliorare il loro lavoro. L'obiettivo è gradualmente portare i programmatori a migrare a .NET in modo indolore.

VISUAL STUDIO 2005 ULTIMA TAPPA

Microsoft ha appena rilasciato un'altra versione preview di Visual Studio 2005, .NET 2.0 e SQL Server 2005. I programmatori attribuiscono a questi tre progetti un'importanza estremamente rilevante. A fare la parte del leone è sicuramente .NET 2.0 che include novità di una certa importanza che rappresentano un notevole salto di qualità alla già rivoluzionaria tecnologia .NET. Non sono da meno gli altri due tool che compongono il tritti-co: Visual Studio 2005 e SQL Server 2005. Il primo sarà il naturale supporto programmatico per .NET 2.0, il secondo è il contenitore di dati per eccellenza sulla base del quale si fondano moltissime applicazioni soprattutto professionali in ambito Windows. Al di la delle note tecniche che riempiono di contenuto le nuove versioni, è importante notare che queste preview per la prima volta utilizzano la licenza GoLive che consente agli sviluppatori di utilizzare questi software per la produzione di applicazioni destinate alla distribuzione eccetto che per il framework 2.0.

INBox L'esperto risponde...

Testare il DNS

Buongiorno cari ioProgrammatori. Da qualche tempo sto cercando di configurare il BIND per una serie di domini che mantengo in multihoming sul mio indirizzo IP fisso di casa. Non riesco ancora a capire se ho configurato bene il DNS, perché in alcuni casi mi accorgo che viene risolto correttamente, in altri no. C'è un modo per capire se si tratta di un problema di rete o di diffusione del dns o un problema di configurazione?

Antonio

Antonio, un buon modo per chiarirsi ogni dubbio è consultare i siti *http://www.dnsstuff.com* e *http//www.dnsreport.com* che mettono a disposizione dei test standard e provati per la verifica della corretta configurazione del dns.

Non riesco a installare l'estensione PDF per PHP

Da più di un anno compro la vostra rivista ed è veramente speciale. Grazie ai vostri cd ho installato Apache, php5 e Mysql. Grazie ad un vostro articolo sul numero 84 ho configurato Apache Mysql e Php5 e tutto è andato bene. Purtroppo quando cerco di decommentare la linea di codice nel file php.ini corrispondente alla libreria php_pdf.dll per creare un documento pdf, viene fuori una finestra che mi dice di non riuscire a trovare il modulo php_pdf.dll. Mi potreste aiutare a risolvere questo problema?

Aurelio

Gentile Aurelio, tutte le estensioni di PHP si trovano nella directory ext situata sotto la radice di PHP. Per prima cosa controlli che la dll in questione sia presente in quella directory. Apra poi il file php.ini e controlli la seguente riga:

```
extension_dir = "./ext"
```

Questa istruzione avverte PHP di cercare le estensioni nella directory ext immediatamente sotto il percorso di installazione di PHP. Infine decommenti la linea

```
;extension= php_pdf.dll
```

Riavvii Apache. Se tutto andrà a buon fine, non riceverà errori. Può effettuare un successivo controllo creando un file info.php contenente le seguenti istruzioni:

```
<?
phpinfo();
?>
```

Puntando il browser all'indirizzo *http://localhost/info.php* otterrà una pagina contenente tutte le caratteristiche di installazione del PHP. Nella sezione delle estensioni comparirà anche quella relativa ai PDF.
Se così non fosse, qualcosa è andato storto. Dovrà adattare la linea extension_dir al path della sua installazione di PHP.

Quando e perché installare un firewall?

Gentile redazione. Ho visto che con il SP2 di Windows viene installato di default un firewall. Se, per caso, lo disabilito, il sistema mi avverte che sono esposto a rischi. Ma a quali rischi sono esposto? Da cosa mi protegge il firewall?

Ho letto il vostro articolo sugli attacchi Ddos e, dato che spesso mi capita che il computer rallenti quando sono connesso a Internet, vorrei capire se secondo voi può dipendere da un attacco Ddos. Con il Firewall aperto non dovrei essere protetto?

Gianluca

Un firewall è un software che filtra i pacchetti quando arrivano alla macchina su cui è installato. Filtrare i pacchetti significa che la scheda li riceve, ma non lascia che vengano elaborati dal software a cui sono destinati. Ad esempio: un pacchetto destinato alla porta 80 potrebbe essere bloccato all'ingresso dal firewall, questo significa che il vostro Web Server, ovvero il servizio che gira sulla porta 80 non riceverà mai quel pacchetto e non elaborerà mai i comandi che conteneva.
Va da sé dunque, che un FireWall è utile per far sì che pacchetti con istruzioni pericolose non arrivino a determinati servizi. Se per esempio state chattando, sulla vostra macchina si apriranno delle porte. Un malintenzionato potrebbe mandare dei pacchetti su queste porte tentando di sfruttarle per eseguire istruzioni pericolose o non consentite. Per cui un firewall è utile per far sì che i pacchetti non giungano a destinazione. È meno utile in caso di attacchi Ddos. Di fatto un attacco Ddos è composto in modo che l'attaccante o gli attaccanti tentino di saturare le risorse della vostra macchina. Ad esempio: se la vostra macchina ricevesse 1.000.000 di ping in brevissimo tempo, dovrebbe elaborare 1.000.000 di pacchetti, con un conseguente rallentamento. Se ci fosse un firewall installato, sarebbe il firewall a dover gestire i pacchetti in arrivo. Chiaramente questa attività rallente-

rebbe il Sistema Operativo. In conclusione: installare un firewall sulla propria macchina è importante per non lasciare passare pacchetti potenzialmente pericolosi, abbastanza inutile invece nel caso di attacchi DDos. Se si temono questo genere d'attacchi è utile nascondersi dietro un router; in tal caso la vostra navigazione su Internet sarà lenta o inattuabile, ma localmente il vostro sistema non subirà eccessivi rallentamenti.

Programmazione di IRC

Cari amici di ioProgrammo, sono uno studente di ingegneria informatica al primo anno. Passo molto tempo in IRC e mi piacerebbe avere un canale tutto mio. Ho visto che i canali sono spesso mantenuti da bot. Vorrei capire come sono fatti questi bot e come funzionano. Mi spiegate qualcosa in più?

Michele

Caro Michele, i bot irc sono ovviamente tenuti in piedi da un software che emula una persona umana, sfrutta i tradizionali protocolli ed occupa un canale irc. La differenza sostanziale con le persone umane è che sono programmabili. Dove per programmabili si intende che possono compiere azioni specifiche in reazione ad eventi specifici. Se per esempio un utente non gradito entra in un canale, sono in grado di riconoscerlo attraverso la sua mask ed espellerlo. Possono mantenere una lista degli utenti non graditi, oppure una lista di utenti a cui passare l'op quando entrano in un canale. Chiaramente, qualunque software conforme agli standard Irc, potrebbe essere programmato per assolvere alle funzioni che abbiamo appena esposto, tuttavia c'è uno standard che si è affernato nel tempo ed è l'Eggdrop – *http://www.geteggdrop.com* – Si tratta di un software scritto in C++, OpenSource e liberamente disponibile, per cui studiabile anche per comprendere come questi meccanismi vengono gestiti. Altra importante funzionalità di un eggdrop è che può essere esteso tramite script realizzabili in TCL. Per cui ad esempio puoi installare il tuo Eggdrop che si connette automaticamente a Irc e compie le azioni di base, per poi estenderlo usando tcl per compiere azioni specifiche. Si stanno molto diffondendo negli ultimi tempi, script che realizzano quiz su canali irc.

IIS6 e Apache non funzionano insieme sulla stessa porta

Ho appena comperato una macchina per fare hosting. Dato che la maggior parte dei software: PHP, MySQL girano anche in ambiente Windows mentre il contrario, cioè ASP e ASP.NET su ambiente Linux è un po' più complesso, ho deciso di acquistare una macchina Windows 2003 con IIS6 installato sopra. Nonostante questo vorrei tenere attivo anche Apache. Ho provato in tutti i modi ma non c'è verso di fare girare IIS e Apache sulla stessa porta. Conoscete qualche trucco per farlo?

Fernando

È impossibile fare girare due servizi sulla stessa porta e sullo stesso IP. È invece possibile fare girare due servizi che offrono identici servizi su porte differenti ad esempio la 80 per IIS e la 8080 per Apache. Tuttavia questo non sembra essere il tuo obiettivo. È anche possibile fare girare due servizi identici sulla stessa porta ma su indirizzi IP differenti. Questa operazione presenta qualche problema. Nel caso di IIS6 poi è assolutamente incompatibile. IIS6 usa infatti un meccanismo chiamato SocketPooling che tralasciando i dettagli più tecnici, impedisce di fatto di fare girare Apache e IIS6 sulla stessa porta anche se su IP differenti. Sarebbe necessario disabilitare il SocketPooling su IIS6 ma questa operazione sconsigliata rallenterebbe di molto il server.

A chi spedire la posta?

Alla nostra redazione arriva spesso un considerevole numero di email riguardante temi specifici. Per consentirci di rispondere velocemente e in modo adeguato alle vostre domande abbiamo elaborato una FAQ – Frequently Ask Question – o risposte alle domande frequenti in italiano, di modo che possiate indirizzare le vostre richieste in modo mirato.

Problemi sugli abbonamenti

Se la tua domanda ha a che fare con una delle seguenti:

- Vorrei abbonarmi alla rivista, che devo fare?
- Sono un abbonato e non ho ricevuto la rivista, a chi devo rivolgermi?
- Sono abbonato ma la posta non mi consegna regolarmente la rivista, a chi devo rivolgermi?

Contatta **abbonamenti@edmaster.it** specificando che sei interessato a ioProgrammo. Lascia il tuo indirizzo email e indica il numero dal quale vorresti far partire l'abbonamento. Verrai contattato al più presto. Oppure puoi chiamare lo **02 831212**

Problemi sugli allegati

Se riscontri un problema del tipo:

- Ho acquistato ioProgrammo ed il Cdrom al suo interno non funziona. Chi me lo sostituisce?
- Ho acquistato ioProgrammo ma non ho trovato il cd/dvd all'interno, come posso ottenerlo?
- Vorrei avere alcuni arretrati di ioProgrammo come faccio?

Contatta **servizioclienti@edmaster.it**
Non dimenticare di specificare il numero di copertina di ioProgrammo e la versione: con libro o senza libro. Oppure telefona allo **02 831212**

Assistenza tecnica

Se il tuo problema è un problema di programmazione del tipo:

- Come faccio a mandare una mail da PHP?
- Come si instanzia una variabile in c++?
- Come faccio a creare una pagina ASP.NET

o un qualunque altro tipo di problema relativo a tecniche di programmazione, esplicita la tua domanda sul nostro forum: **http://forum.ioprogrammo.it**, uno dei nostri esperti ti risponderà. Le domande più interessanti saranno anche pubblicate in questa rubrica.

Problemi sul codice all'interno del CD

Se la tua domanda è la seguente

- Non ho trovato il codice relativo all'articolo all'interno del cd

Consulta la nostra sezione download all'indirizzo **http://cdrom.ioprogrammo.it**, nei rari casi in cui il codice collegato ad un articolo non sia presente nel cdrom, senza dubbio verrà reso disponibile sul nostro sito.

Idee e suggerimenti

Se sei un programmatore esperto e vuoi proporti come articolista per ioProgrammo, oppure se hai suggerimenti su come migliorare la rivista, se vuoi inviarci un trucco suggerendolo per la rubrica tips & tricks invia una email a **ioprogrammo@edmaster.it**

NEWSGROUP

Le informazioni nella Rete

Direttamente dal forum di ioProgrammo le discussioni più "Hot" del momento

ADO.NET

Come mettere una intestazione di colonna in ADO.NET

Volevo sapere se esiste una proprietà per dare il nome anche alle righe come alle colonne. Ad esempio:

```
               |   Titolo    |   Titolo
               |  Colonna1   |  Colonna2
---------------|-------------|-------------
TitoloRiga     |   Valore    |   Valore
---------------|-------------|-------------
```

Rappresentazione Standard:

```
               |  Prodotto   |   Descr.
---------------|-------------|-------------
|      1       |     PC      |
---------------|-------------|-------------
```

http://forum.ioprogrammo.net/thread.php?threadid=5463&boardid=27

sklero

Risponde Francesco Smelzo

Per fare bene dovresti:

- alterare la fonte dei dati in modo che la prima colonna del *DataTable* contenga il nome di riga.
- Create una estensione della classe *DataGridColumnStyle* da applicare alla prima colonna in modo che il nome di riga non venga in una textbox, non sia editabile e magari venga disegnato con uno sfondo tipo bottone.
- Prima dell'associazione *dbGrid/datasource* devi creare un nuovo *DataGridTableStyle* nella collection *TableStyles* della dataGrid rimappando i campi del DataTable e associando quello della prima riga alla classe di estensione di sui sopra.

In realtà, questo sarebbe il metodo migliore, ma se i tempi sono stretti puoi anche, visti i tempi che hai, utilizzare il controllo FlexGrid di vb6 direttamente da VB.NET.

Un modo per farlo è il seguente:

In Visual Studio:

1. Clicchi con il tasto destro sul pannello *strumenti*
2. *Aggiungi/rimuovi elementi*
3. Nella finestra scegli *"Componenti Com"* e quindi selezioni *"Microsoft Flex Grid"*

Ti dovrebbe quindi apparire il controllo sul *pannello strumenti*.

Lo usi come in vb6, ricordati però che i dati non gli devono arrivare direttamente da ADO.NET ma dal "vecchio" ADODB che puoi ancora invocare da VB.NET anche con il classico:

```
dim conn as object=CreateObject(
        "ADODB.CONNECTION") ecc...
```

Visual Basic

Chiudere un processo in VB6

Ciao a tutti, vorrei da un programma scritto in vb6 chiudere un altro programma sempre scritto in vb6. Ho utilizzato alcune api, trovate in internet, ma... non funzionano non mi chiude il processo. Come posso fare?

http://forum.ioprogrammo.net/thread.php?threadid=5446&boardid=13

Toyota

Risponde SimoneVB

Ci sono diverse soluzioni possibili. Quella che ti consiglio fa uso delle classi di WMI (argomento trattato recentemente da ioProgrammo).

Da VB puoi utilizzare Windows Management Instrumentation riferendoti alla libreria "Microsoft WMI Scripting v1.1 Library" oppure in late-binding come nell'esempio che segue:

```
Sub TerminateProcess(ByVal ProcessName
                              As String)
'***********************************
' Termina un processo
'***********************************
Dim oLoc
Dim oServ
Dim oObjectSet
Dim oProc
Dim sMsg As String

' query WQL
Const sWQL As String = "SELECT * FROM
        Win32_Process WHERE Name =
                        'notepad.exe'"

' oggetto locator
Set oLoc = CreateObject(
        "WbemScripting.sWbemLocator")
Set oServ = oLoc.ConnectServer(".",
                          "root\cimv2")
Set oObjectSet = oServ.ExecQuery(sWQL)
For Each oProc In oObjectSet
  If MsgBox("Terminare " & oProc.Caption _
      & " ?", vbExclamation + vbYesNo,
      "Termina processo") = vbYes Then
                      oProc.Terminate
Next
Set oProc = Nothing
Set oObjectSet = Nothing
Set oServ = Nothing
Set oLoc = Nothing
End Sub
```

Chiaramente, devi controllare che WMI sia installato e che il notepad sia in esecuzione. In realtà potresti anche parametrizzare la query per rendere tutto più portabile.

Nominare sequenzialmente i file in VB

Ho realizzato un engine per scacchi. Il programma è in test, per cui creo un file testuale di debug. Il banale problema che ho è che vorrei dare al file il nome del giocatore contro cui gioca, ma con un numero sequenziale creato dal programma: cioè se esiste pippo03 vorrei nominare il file pippo04. Al momento, l'unico modo che conosco per sapere se un file esiste, è eseguire la open:

1)con "on error" il file non c'è e quindi apro il file col nome dell'avversario
2)il file c'è, quindi devo dare un altro nome, incrementando 03, 04, 05..

È laborioso anche perché non so quanti ce ne possano essere di file..di pippo.. Chi mi suggerisce un metodo diverso?

http://forum.ioprogrammo.net/thread.php?threadid=5426&boardid=13

Dario 70

Risponde Hyde

Se non vuoi scomodare il FileSystem per non provocare un overhead del sistema e comunque per utilizzare una sintassi più semplice ed immediata, puoi utilizzare qualcosa di questo genere, che ti semplifica di molto la vita.

```
Private Function FileExist(ByVal nfile As String) As Boolean
 On Error Resume Next
 FileExist = (GetAttr(nfile) And vbDirectory) = 0
 End Function
```

Chiaramente la funzione restituisce un booleano a seconda che il file esista oppure meno. Il resto è abbastanza semplice. In questo modo non devi aprire e chiudere un file tutte le volte che vuoi controllarne l'esistenza. La conseguenza immediata è un minore rallentamento dell'applicazione.

Java

Chiamare un WebService da JSP

**Salve a tutti, sto tentando di richiamare da una pagina jsp di un sito sotto tomcat, un web service da me sviluppato, funzionante e pubblicato fra l'altro sulla stessa macchina, sotto tomcat/axis.
Attualmente dalla pagina jsp chiamo una classe bean che ha un metodo al cui interno ho messo il codice che chiama il web service (Call call = new call etc...): non vuole funzionare mi dà errore Jasper e Servlet.
Il bean con metodi banali funziona.**

http://forum.ioprogrammo.net/thread.php?threadid=5363&boardid=18

webmasterit77

Risponde doc

Io di solito creo lo stub dal WSDL e poi da una pagina jsp lo richiamo in questo modo

```
<%
try
{
  StubServizio stub = new StubServizio();
  result = stub.sayHello("Ciao");
}

catch (Exception e)
{
  result = "Problemi!!!";
}
%>
```

Per creare lo stub puoi usare *wsdl2-java* che crea automaticamente tutto.

Chiarimenti su "a parola di java"

**Ciao a tutti, trovo molto interessante l'articolo proposto da numero di Aprile di Io Programmo. Ad un certo punto però si consiglia di passare il valore -mx312m alla JVM affinché l'applicazione di riconoscimento vocale possa funzionare.
Il mio problema sarà semplicissimo per i più: come si passa tale valore alla JVM?**

http://forum.ioprogrammo.net/thread.php?threadid=5379&boardid=18

Sonique

Risponde Doc

Semplicemente devi passare quel valore quando lanci l'applicazione che usa Sphinx4, in questo modo

```
java -mx312m MioProgCheUsaSphinx4
```

C++

Come gestire una Sequenza di Label in C#

Salve a tutti. Ho tante label etichettate con il nome:

lblAb1
...
...
lblAb30

Se volessi ricorrere ad un ciclo for per impostare un parametro come posso fare?

http://forum.ioprogrammo.net/thread.php?threadid=5388&boardid=29

Merlino666

Risponde Antonio Pelleriti

Puoi usare un foreach per effettuare un ciclo su tutti i controlli e verificare quali di essi sono di classe Label.
Se il controllo corrisponde alla classe desiderata puoi settarne le proprietà secondo le tue esigenze.
Ad esempio, così si imposta a rosso il BackColor di tutte le Label:

```
Label lab;
foreach(Control c in this.Controls)
{
    if(c is Label)
    {
        lab=(Label)c;
        lab.BackColor=Color.Red;
    }
}
```

Gestire la Cam del Cellulare

La teoria e la tecnica per creare una completa applicazione per catturare immagini e video dal telefonino e inviarle a un server su Internet. Un ottimo punto di partenza per creare un blog fotografico

L'idea di questo articolo è molto semplice. Possedete un cellulare di ultima generazione in grado di realizzare e salvare filmati? Molto semplicemente costruiremo un'applicazione personalizzata che vi consentirà di girare una scena con il cellulare e salvarla direttamente su un server in internet. Non ci crederete, ma questo tipo di meccanismo si sta diffondendo con il nome di MoBLOG, si tratta di blog dove invece di postare testi si inviano foto e filmati, appunti di vita quotidiana.

COME FUNZIONERÀ L'APPLICAZIONE

Possiamo pensare che, una volta avviata l'applicazione, essa ci mostri ciò che la telecamera riprende. Una serie di comandi ci aiuteranno ad individuare le azioni da intraprendere. Alla voce "Opzioni", sulla barra dei comandi del telefono, corrisponderà un menu a tendina, in cui vengono visualizzate le voci "Cattura foto", "Registra video". Quando si preme il tasto "fire" relativo alla voce "Cattura foto" il display cattura un'immagine e la visualizza. A questo punto avremo due opzioni: "Invia al repository" o "Indietro". Nel primo caso potremo inserire in un TextField il nome del file in cui salveremo i dati sul server e premendo "Invia" una pop-up ci avvertirà che il trasferimento è in corso o in caso di problemi di rete ci inviterà a riprovare più tardi. Comunque si tornerà alla vista sulla camera attiva. Se desideriamo registrare un video, continueremo a mantenere la vista sulla camera attiva e l'unica azione che potremo effettuare sarà lo "Stop registrazione". Una volta fermata la registrazione viene visualizzato l'ultimo frame del video e il comportamento è uguale al caso precedente, con le due opzioni "Invia al repository" o "Indietro".

Fig. 1: Cliccando su options apparirà il menu per inviare la foto

COME INIZIARE

Il primo passo per sviluppare un'applicazione MMAPI è individuare i tools appropriati. Il Sun J2ME Wireless Toolkit include il supporto per MIDP 2.0 e MMAPI 1.1 e fornisce alcuni emulatori standard, è necessario scaricarlo all'indirizzo: http://java.sun.com/products/j2mewtoolkit/ In alternativa si può utilizzare il Nokia Developer's Suite for J2ME disponibile sul Forum Nokia www.forum.nokia.com, il sito di Nokia per gli sviluppatori. Questa suite contiene gli emulatori per molti modelli di cellulari Nokia e le SDK per poter sviluppare applicazioni utilizzando API proprietarie di Nokia. Gli emulatori possono essere inclusi anche nel Wireless Toolkit. Emulatori per cellulari di altre marche (ad esempio Sony Ericsson) si possono scaricare solitamente dai siti della casa produttrice e poi si possono includere facilmente nel Wireless Toolkit: basta copiare la directory relativa all'emulatore in *C:\WTK22\wtklib\devices* se *C:\WTK22* è la root dove è installato il Wireless Toolkit.

REQUISITI

Conoscenze richieste
Basi di Java

Software
J2ME

Impegno

Tempo di realizzazione

FACCIAMOLO IN JAVA

Le applicazioni Java che girano su dispositivi MIDP sono chiamate MIDlet. Una MIDlet è costituita da almeno una classe derivata dalla classe astratta javax.microedition.midlet.MIDlet. La struttura di una MIDlet è la seguente:

```
public class MyMIDlet extends MIDlet {
CameraMIDlet( ) {}
protected void startApp( ) throws
                        MIDletStateChangedException {}
protected void pauseApp( ) {}
protected void destroyApp(boolean unconditional)
                throws MIDletStateChangedException {}
}
```

Nel metodo startApp() deve essere implementata la logica di esecuzione della MIDlet. Tale metodo viene richiamato subito dopo che è stato invocato il costruttore ed è stata inizializzata la MIDlet. La piattaforma MIDP può mettere la MIDlet in uno stato di pausa, se, ad esempio, durante la sua esecuzione arriva una telefonata. In questo caso viene chiamato il metodo pauseApp(), che va quindi implementato in modo da sospendere l'esecuzione. Analogamente deve essere implementato il metodo destroyApp () che deve rilasciare tutte le risorse allocate, stoppare i thread in backgroud e chiudere l'applicazione.

MOBILECAMERA MIDLET

Per quello che ci riguarda la nostra Midlet si chiamerà MobileCameraMIDlet e come gia detto estenderà la classe astratta MIDlet. Nel suo metodo startApp ( ) vengono create le instanze dei due oggetti CameraCanvas e SendCanvas e viene chiamato il metodo start ( ) di CameraCanvas che avvierà il controllo sulla camera. MobileCameraMIDlet, in quanto oggetto principale dell'applicazione, in qualche modo viene utilizzato anche per coordinare le relazioni tra gli altri due oggetti. In particolare, volendo, che una volta catturata una foto, essa sia visualizzata sul display fino a quando non si intraprende una nuova azione (invio della foto al server o ritorno alla camera attiva), MobileCameraMIDlet riceve la foto catturata da CameraCanvas e la passa a SendCanvas che la visualizza sul display ed eventualmente la invierà al server di repository:

```
public void cameraCanvasImageCaptured(
                                    byte[]imageData) {
   cameraCanvas.stop();
   sendCanvas.setImage(imageData);
   Display.getDisplay(this).setCurrent(displayCanvas);
}
```

Analogamente quando termina la registrazione del video, il display visualizzerà l'ultimo frame del video ripreso, perciò SendCanvas passa a MobileCameraMIDlet l'ultimo frame e l'intero video. SendCanvas visualizzerà sul display l'ultimo frame ed eventualmente invierà il video al server di repository:

```
public void cameraCanvasVideoCaptured(
                    byte[]imageData, byte[]videoData) {
   cameraCanvas.stop()
   sendCanvas.setVideo(imageData,videoData);
   Display.getDisplay(this).setCurrent(sendCanvas);
}
```

In ultimo MobileCameraMIDlet si occupa di gestire la visualizzazione di messaggi di tipo pop-up che compaiono sul telefono per avvertire l'utente di una particolare situazione, che può essere una semplice informazione, un warning o un errore. L'oggetto che vuole utilizzare questo metodo deve passargli solamente il testo del messaggio da stampare a video:

```
public void displayAlert(String body) {
   Alert alert = new Alert("", body, null, AlertType.INFO);
   alert.setTimeout(3000);
   Display.getDisplay(this).setCurrent(alert);
 }
```

GESTIONE DELLA CAMERA

Tutto è affidato a CameraCanvas che estende l'oggetto Canvas. La classe Canvas è la classe base per scrivere applicazioni in cui bisogna disegnare sul display o comunque averne un controllo a livello grafico. Il metodo paint ( ) è dichiarato astratto e deve essere implementato dallo sviluppatore. Nella nostra realizzazione, nel caso in cui il telefono non supporti la cattura video nel metodo paint ( ) andremo a colorare il display di nero, mostrando dei messaggi d'errore, altrimenti coloreremo lo sfondo di giallo. La classe CameraCanvas viene utilizzata per collegare le azioni della camera del telefono al suo display. All'interno di questa classe sarà quindi implementato anche ciò che riguarda le funzionalità specifiche richieste dalla nostra applicazione: cattura di una foto e registrazione di un video. Vediamo quali sono le operazioni che vengono effettuate nel costruttore di CameraCanvas. Il primo passo è quello di definire dei comandi (Command) che corrispondono alle azioni "Cattura foto", "Registra video", "Stop registrazione", "Esci". Ciò significa che CameraCanvas implementa l'interfaccia CommandListener e definisce il metodo commandAction(Command c, Displayable d).

```
captureCommand = new Command(
                    "Cattura foto",Command.SCREEN,1);
recordCommand = new Command(
```

```
                              "Registra video",Command.SCREEN,1);
stopRecordCommand = new Command(
                        "Stop registrazione",Command.SCREEN,1);
exitCommand = new Command(
                              "Esci",Command.EXIT,1);
addCommand(captureCommand);
addCommand(recordCommand);
addCommand(exitCommand);
setCommandListener(this);
```

Il comando "Stop registrazione" verrà aggiunto soltanto successivamente, perché desideriamo che esso compaia soltanto durante la registrazione del video. Il secondo passo eseguito nel costruttore consiste nel connettere l'applicazione al dispositivo camera del telefono. L'oggetto di cui abbiamo bisogno ce lo forniscono le MMAPI ed è il Player. Esso viene ottenuto tramite un Manager. Uno speciale locator, "capture://video" indica che l'immagine sarà catturata utilizzando un formato di default.:

```
player = Manager.createPlayer("capture://video");
```

Se il dispositivo non supporta la cattura video con le caratteristiche richieste, viene lanciata una MediaException. Occorre poi che il Player sia nello stato realized per ottenere le risorse di cui ha bisogno per catturare immagini:

```
player.realize ( );
```

Per visualizzare sul display del telefono ciò che la camera riprende, utilizziamo un oggetto derivato dall'interfaccia Control, il VideoControl. Per ottenere un VideoControl basta chiederlo al Player:

```
videoControl = (VideoControl)(player.getControl(
                                        "VideoControl"));
```

A questo punto inizializziamo il VideoControl secondo la modalità con cui vogliamo che il flusso ripreso venga visualizzato:

```
videoControl.initDisplayMode(
                  VideoControl.USE_DIRECT_VIDEO,this);
```

Nel costruttore abbiamo quindi definito e preparato

LA NOSTRA APPLICAZIONE VISTA IN DIRETTA

START!

1 Avviata l'applicazione, la camera viene attivata e viene visualizzato ciò che essa riprende. Useremo i menu per scegliere le operazioni da compiere

FOTO O VIDEO?

2 Premendo il tasto relativo allo voce "Opzioni" compare un menu a tendina: dobbiamo fare la nostra scelta: "Cattura foto" o "Registra video".

REGISTRA VIDEO

3 Dopo aver scelto "Registra video" la camera resta attiva, l'applicazione registra il video, in attesa dello "Stop registrazione".

CATTURA FOTO

4 Dopo aver scelto "Cattura foto" l'immagine viene visualizzata. Quando viene stoppata la registrazione viene visualizzato l'ultimo frame.

SALVA CON NOME...

5 Se decidiamo di inviare l'immagine o il video al server prima dobbiamo fornire il nome con il quale essa sarà salvata nel repository.

INVIO AL REPOSITORY

6 A questo punto inviamo l'immagine e attendiamo che essa venga trasferita. Intanto l'applicazione torna allo stato iniziale: vista dalla camera.

all'uso tutte le risorse di cui abbiamo bisogno. Nel metodo start () non faremo altro che avviare il Player e rendere visibile sul display il flusso ripreso dalla camera:

```
player.start();
videoControl.setVisible(true);
```

Al contrario nel metodo stop ( ) stoppiamo il Player e rendiamo invisibile quanto ripreso dalla camera:

```
videoControl.setVisible(false);
player.stop();
```

Ora siamo in grado di vedere sul telefono ciò che la camera riprende. Il successivo step è intraprendere un'azione. Abbiamo detto che l'azione è legata ad un comando. Ad esempio, premendo il tasto relativo alla voce "Cattura foto" viene catturata una foto. Codificando quanto appena detto all'interno del commandAction(Command c, Displayable d), abbiamo:

```
public void commandAction (Command c,
                                   Displayable d) {
   if(c == captureCommand) {
      captureFoto(); }
   ...
```

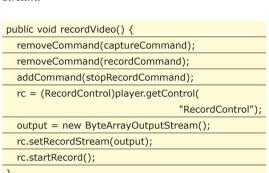
Il metodo captureFoto ( ) è quello che in effetti cattura la foto, la memorizza in un array di byte e la passa alla midlet (che a sua volta la passa a SendCanvas in modo che sia impostata come sfondo):

```
public void captureFoto() {
   if(player != null) {
      try {
         imageData = videoControl.getSnapshot(
            "encoding=jpeg&width=640&height=480");
         midlet.cameraCanvasCaptured(imageData);
      } catch(MediaException me) {
         message1 = "MediaException:";
         message2 = me.getMessage();} }
}
```

La cattura vera e propria viene effettuata dal metodo *getSnapshot(String imageType)*. Le caratteristiche e il formato delle immagini sono specificate nella stringa imageType. I formati supportati possono essere richiesti con la chiamata *System.getProperty ("video .snapshot.encodings")*. Nel nostro caso abbiamo richiesto delle foto in formato jpeg, 640x480. Come già detto, una volta catturata l'immagine essa viene consegnata, passando per la classe MobileCameraMIDlet, a SendCanvas che la setta come sfondo e dà la possibilità di inviarla. Vediamo ora ciò che succede quando si preme il tasto relativo alla voce "Registra video". Nel metodo CommandAction questa volta avremo:

```
if (c == recordCommand) {recordVideo();}
```

Il metodo *recordVideo ( )* innanzitutto rimuove i comandi *"Cattura foto"* e *"Registrazione video"* ed aggiunge *"Stop registrazione"*. Successivamente chiede al Player il Control di tipo *RecordControl* ed inizia la memorizzazione dei dati su un *ByteArrayOutputStream:*

```
public void recordVideo() {
   removeCommand(captureCommand);
   removeCommand(recordCommand);
   addCommand(stopRecordCommand);
   rc = (RecordControl)player.getControl(
                                "RecordControl");
   output = new ByteArrayOutputStream();
   rc.setRecordStream(output);
   rc.startRecord();
}
```

L'unica azione che possiamo ora intraprendere è "Stop registrazione":

```
if (c == stopRecordCommand) {stopRecording();}
```

Il metodo *stopRecording ( )* interrompe la registrazione, cattura l'ultimo frame del video, trasferisce i dati da un ByteArrayOutputStream ad un ByteArray e consegna ultimo frame e video, passando per la classe MobileCameraMIDlet, a SendCanvas che setterà l'ultimo frame come sfondo e darà la possibilità di inviare il video.

IL FORMATO DEL VIDEO

Il formato del video catturato sarà proprio quello specificato nel locator. Perciò se non forniamo altri dettagli il video catturato avrà dimensione pari a quelle del display e codifica uguale a quella normalmente utilizzata dal dispositivo (ad es. mpeg). Quindi, se, ad esempio, il nostro telefono è di tipo QCIF (display 176x144 pixel), ma vogliamo un video i cui frame siano ad esempio 640 x 480, il locator sarà:

```
player = Manager
.createPlayer("capture:
         //video?widht=
         640&height=480");
```

con la stessa sintassi utilizzata per le url.

INVIARE I DATI

SendCanvas è la classe utilizzata per l'invio dei dati al repository server. Essa estende la classe Canvas perché viene utilizzata anche per mostrare come sfondo sul display una data immagine. Nel costruttore di SendCanvas creiamo i comandi che ci permetteranno di passare alla schermata di inserimento del nome del file (per poi inviare i dati) o tornare alla camera attiva: "Invia al repository" e "Indietro":

```
backCommand = new Command(
                     "Indietro",Command.BACK,1);
sendCommand = new Command(
         "Invia al repository",Command.SCREEN,1);
sendDataCommand = new Command(
                     "Invia",Command.SCREEN,1);
```

```
addCommand(backCommand);
addCommand(sendCommand);
setCommandListener(this);
```

Abbiamo poi i due metodi che vengono chiamati da MobileCameraMIDlet quando da CameraCanvas è stata catturata una foto o registrato un video: setImage (byte[] imageData) e setVideo(byte[] imageData, byte[] videoData). Il primo crea un'immagine dai dati ricevuti e setta il content type dei dati a "immagine". Il secondo crea un'immagine da imageData, memorizza il video contenuto in videoData e setta il content type dei dati a "video":

```
public void setImage (byte[] imageData) {
   this.imageData = imageData;
   image = Image.createImage(
                      imageData,0,imageData.length);
   sendContent = 'i'; // content type: immagine }
public void setVideo(byte[] imageData, byte[]
                                       videoData) {
   this.imageData = imageData;
   image = Image.createImage(
                      imageData,0,imageData.length);
   this.videoData = videoData;
   sendContent = 'v'; // content type: video
}
```

In entrambi i casi, a questo punto il metodo paint ( ) si accorge di avere un'immagine a disposizione e la disegna sullo schermo:

```
public void paint (Graphics g) {
   g.setColor(0x0000FFFF); // cyan
   g.fillRect(0,0,getWidth(),getHeight());
   if(image != null) {
      g.drawImage(image,getWidth(),getHeight(),
             Graphics.VCENTER|Graphics.HCENTER);}
}
```

Anche il funzionamento di questa classe, come di CameraCanvas, è basato sulle azioni legate ai comandi. Alla pressione del tasto corrispondente a "Invia al repository" avviene il passaggio alla schermata per l'inserimento del nome del file:

```
if (c == sendCommand) {
   fileNameTF = new TextField("Salva come...",
                            "",15,TextField.ANY);
   saveFileForm = new Form("MobileCamera");
   saveFileForm.append(fileNameTF);
   saveFileForm.addCommand(backCommand);
   saveFileForm.addCommand(sendDataCommand);
   saveFileForm.setCommandListener(this);
   Display.getDisplay(midlet).setCurrent(saveFileForm);
}
```

In pratica viene costruito e visualizzato un Form contenente un TextField, in cui si inserisce il nome del file (massimo 15 caratteri). Al Form viene aggiunto il comando corrispondente ad "Invia" la cui azione è:

```
if (c == sendDataCommand) {
   switch (sendContent) {
      case 'i': // image
      data = imageData;
      break;
      case 'v': // video
      data = videoData;
      break; }
   String fileName = fileNameTF.getString();
   int numSpace = 15 - fileName.length();
   for (int k=0; k < numSpace; k++) {
      fileName = fileName + " "; }
   byte[] fileNameByte = fileName.getBytes();
   dataMod = new byte [data.length + 1 +
                             fileNameByte.length];
   dataMod[0] = (byte)sendContent;
   for (int i=0; i < fileNameByte.length; i++) {
      dataMod[i+1] = fileNameByte[i]; }
   for (int j=0; j < data.length; j++) {
      dataMod[j + fileNameByte.length + 1] = data[j];}
   sendData (dataMod);
}
```

I dati da inviare non sono soltanto quelli relativi all'immagine o al video. In un array di byte viene infatti inserito come primo byte il carattere identificativo del content type ("i" per immagine oppure "v" per video), quindi il nome del file destinazione e poi il contenuto vero e proprio. Infine, finalmente i dati vengono spediti, attraverso il metodo sendData (byte[] data). Supponiamo di trasferire i dati tramite socket. Otteniamo un oggetto SocketConnection utilizzando il metodo Connector.open (String uri).
La sintassi della *url* deve essere conforme alla seguente: "socket://host:port". Ora che abbiamo a disposizione una connessione ci occorre un canale su cui scrivere i dati: si tratta di uno StreamConnection. Vediamo l'implementazione:

```
void sendData (byte[] data) {
   try {
      socketConn = (SocketConnection)Connector.open(
                            "socket://88.88.88.88:88");
      midlet.displayAlert("Attendere...trasferimento in
                                             corso!");
      sc = (StreamConnection) socketConn;
      dos = sc.openDataOutputStream();
      dos.write(imageData);
      dos.flush();
   } catch (IOException ioe) {
      midlet.sendCanvasBack();
      midlet.displayAlert("Trasferimento impossibile...
                               riprovare più tardi");}
```

```
finally {
   try {
      dos.close();
      sc.close();
      socketConn.close();
      midlet.sendCanvasBack();
      midlet.displayAlert("Connessione col server
                                        chiusa!");
   } catch (IOException ioe) {
      midlet.sendCanvasBack();
      midlet.displayAlert("Errore di comunicazione
                                col server!");} }
}
```

REPOSITORY SERVER

Come repository server si può utilizzare una struttura basata sull'application server Apache Tomcat. Il server (lo chiamiamo MultimediaRepository) è composto di due oggetti: la classe RepositoryInit che inizializza il server, e la classe RequestManager che gestisce le richieste del client (invio dati).
RepositoryInit legge da file di configurazione (MultimediaRepository.ini) informazioni quali la porta su cui il server è in ascolto, il path del file di configurazione dei log, la directory dei log, la directory dove salvare le immagini e quella dove salvare i video. Inoltre instanzia un ServerSocket sulla porta specificata dal file di configurazione e si mette in ascolto di nuove connessioni socket stabilite da un client. Infine passa l'oggetto Socket, corrispondente ad una nuova connessione, al RequestManager:

```
ServerSocket serverSocket = new ServerSocket(
                                    SERVER_PORT);
//Mi metto in ascolto di chiamate dai client
while( true ) {
   logger.info("Server in attesa di chiamate da client
                      sulla porta " + SERVER_PORT );
   Socket clientSocket = serverSocket.accept();
   logger.info("Ricevuta chiamata apertura socket da:
                  " + clientSocket.getInetAddress() );
   RequestManager manageRequest = new
                     RequestManager(clientSocket);
   manageRequest.start();
   logger.info("Avviato thread per gestire la
                                   chiamata...");
}
```

Il metodo RequestManager apre un BufferedInputStream sull'InputStream relativo al Socket che gli viene passato e legge i dati. Elaborando i dati ricevuti ottiene le informazioni che gli occorrono per salvare un'immagine col suo nome nella directory relativa alle immagini oppure un video col suo nome nella directory relativa ai video:

```
´÷if (clientSocket.isClosed()) {
   // content type
   char dataType =(char)byteArrayClient[0];
   // file name
   String fileName = "";
   for (int i=1; i <= 15; i++) {
      fileName += (char)byteArrayClient[i]; }
   // immagine o video
   byte[]data = null;
   for (int j=16; j < byteArrayClient.length; j++) {
      data[j-16] = byteArrayClient [j]; }
   String saveDir = "";
   switch (dataType) {
      case 'i': // image
      saveDir = RepositoryInit.IMAGE_DIR;
      break;
      case 'v': // video
      saveDir = RepositoryInit.VIDEO_DIR;
      break; }
   String outputFile = saveDir + File.separator +
                                          fileName;
   FileOutputStream os = new
                       FileOutputStream(outputFile);
   os.write(data);
   os.flush();
}
```

CONCLUSIONI

In questo articolo abbiamo visto come poter realizzare un repository dei contenuti multimediali catturati con il proprio telefono. È un'utile applicazione che ci consente, anche con dispositivi non dotati di una cospicua memoria (interna o esterna), di poter comunque salvare i propri dati. In un prossimo articolo vedremo come effettuare l'operazione inversa, ovvero visionare dal telefono i dati memorizzati su un server e scaricare quelli desiderati.

Vincenzo Viola

CENTRARE LE IMMAGINI

Se abbiamo specificato una cattura video in cui le dimensioni sono diverse da quelle del display del telefono, facciamo in modo, nella visualizzazione, di centrare e ridimensionare ciò che viene ripreso:

```
int canvasWidth = getWidth();
int canvasHeight = getHeight();
int displayWidth = videoControl.getDisplayWidth();
int displayHeight = videoControl.getDisplayHeight();
int x = (canvas Width-displayWidth)/2;
int y = (canvasHeight-displayHeight)/2;
videoControl.setDisplayLocation(x,y);
```

Quando PHP incontra Java

Utilizzare le classi Java direttamente da script PHP. Perché farlo? Prima di tutto per estendere il linguaggio, secondo per consentire a chi conosce Java di riutilizzare le proprie classi anche sul web. Terzo...

La notizia è di quelle che scottano: Zend e Sun hanno annunciato di collaborare all'integrazione di Java con PHP. L'annuncio risale già a qualche tempo fa, ma solo adesso si iniziano a vedere i primi frutti della nuova strategia di collaborazione.
Che cosa vuol dire integrare Java in PHP e quali sono i vantaggi che questa integrazione potrebbe offrire? La prima risposta è abbastanza semplice. Il frutto di questa integrazione è il poter utilizzare le classi e gli oggetti Java da dentro PHP come se fossero oggetti nativi del linguaggio. La risposta alla seconda domanda potrebbe essere più articolata e complessa, i vantaggi sono davvero moltissimi. Prima di tutto, i programmatori Java possono tranquillamente riutilizzare il loro lavoro direttamente sul web senza passare da un server JSP, ma utilizzando tutti i vantaggi di PHP che attualmente è forse il linguaggio più versatile per quanto riguarda le applicazioni Internet Oriented. Il secondo vantaggio immediato è relativo agli sviluppatori PHP che possono sfruttare l'alta integrazione di Java con i sistemi su cui gira una JVM. Qualche esempio? Potreste sviluppare un'applicazione Web che consente ad un utente di immettere un testo e un numero di telefono, la pressione del tasto *"submit"* richiama una classe java che sfrutta *HylaFax* per mandare un fax. Si potrebbe fare lo stesso con gli SMS per esempio. Java si può porre come collegamento fra Internet e l'hardware di una macchina. Oppure ancora è possibile sfruttare la potenza di J2EE, tutto ciò che è possibile fare con Java estende ora le possibilità di PHP. Per certi

COME INIZIARE

Ovviamente dovete avere sul computer una versione funzionante di Apache o IIS con PHP. Gli esempi di questo articolo sono stati testati su una macchina Linux con Apache 1.3, PHP 5 e l'estensione Java Bridge attiva. Per installare Java Bridge su Windows è necessario scaricare il file *php-java-bridge_*-win32-php5.zip* da http://sourceforge.net/projects/php-java-bridge, o prelevarlo dal CD allegato alla rivista. Il file deve essere scompattato nella directory di PHP. Per avviare il bridge basta un doppio click su *c:\php5\JavaBridge.jar.* Per collegare PHP al Java Bridge bisogna copiare i Jar e *php_java.dll* nella directory delle estensioni, infine nel *php.ini* devono essere presenti le seguenti righe:

```
extension = php_java.dll[java]
java.hosts="127.0.0.1:9167"
```

In ambiente Linux l'estensione può e deve essere compilata dai sorgenti, scaricando il tgz da http://sourceforge.net/projects/php-java-bridge, http://sourceforge.net/projects/php-java-bridge Per la compilazione bisogna scompattare i sorgenti e usare

```
phpize && ./configure --
 with-java=/opt /IBMJava2-14 &&
              make CFLAGS="-m32"
su -c 'make install'
<password>
```

Nel *php.ini* bisogna aggiungere

```
extension = java.so[java]
java.socketname=/var/run
          /.php-java-bridge_socket
```

per lanciare il Java Bridge

```
export JAVA_HOME=/usr/lib
          /sun-j2sdk1.5.0
export LD_LIBRARY_PATH=/usr
   /lib/php5 /20041030/usr/lib
  /php5/20041030/RunJavaBridge
   /usr/lib/sun-j2sdk1.5.0/bin/java
 -Djava.library.path=/usr/lib/php5
   /20041030 -Djava.class.path=
 /usr/lib/php5/20041030/JavaBridge
    .jar -Djava.awt.headless=true
  php.java.bridge.JavaBridge /var
 /run/.php-java-bridge_socket 2 ''
```

sostituite ovviamente i path usati nell'esempio con quelli relativi alla vostra installazione

REQUISITI

Conoscenze richieste
basi di PHP

Software
Apache, PHP, PHP Java Bridge

Impegno

Tempo di realizzazione

versi in questo senso Java potrebbe essere quello che il .NET framework è per ASP .NET.

PHP5 VS PHP4

C'è una certa confusione nell'uso delle estensioni Java per PHP. Fino alla versione 4, infatti, la dll o il .so che le rende utilizzabili erano distribuite in bundle con PHP. Se usate PHP4 avrete ancora java.dll disponibile e potrete utilizzarla abbastanza semplicemente decommentando le righe relative nel php.ini. Dalla versione 5 di PHP, le estensioni java non sono più Bundled con PHP, tutto ciò rende necessario i passi di installazione che abbiamo illustrato in questo stesso articolo.

LE MANI NEL CODICE

Non c'è niente di meglio che un bell'esempio per cominciare. Ovviamente inizieremo da qualcosa di semplice:

```
<?php
$system = new Java('java.lang.System');
echo 'Java version=' . $system->getProperty(
                                    'java.version') . '<br />';
echo 'Java vendor=' . $system->getProperty(
                                    'java.vendor') . '<br />';
echo 'OS=' . $system->getProperty('os.name') . ' ' .
             $system->getProperty('os.version') . ' on ' .
             $system->getProperty('os.arch') . ' <br />';
$formatter = new Java('java.text.SimpleDateFormat',
           "EEEE, MMMM dd, yyyy 'at' h:mm:ss a zzzz");
echo $formatter->format(new Java('java.util.Date'));
?>
```

La prima riga instanzia un oggetto *system* di classe *java.lang.system*. A questo punto stiamo gia utilizzando la JVM e le classi di java, non ci resta che richiamare metodi e proprietà tipiche della classe che stiamo usando.
L'esempio è abbastanza chiaro, si usa il metodo *getProperty* per ottenere le informazioni di sistema. Da notare che il comando *echo* è invece un comando PHP. *$system* è un oggetto che fa riferimento alla classe Java, ma all'interno di PHP viene utilizzato come un oggetto nativo. Nelle righe che seguono l'esempio relativo ai dati di sistema, viene instanziato invece un oggetto formatter di classe *java.text.SimpleDateFormat* e successivamente la data viene stampata a schermo tramite il comando *echo*.
Salvando questo file, come java1.php nella radice del mio webserver e puntando il browser all'indirizzo *http://localhost/java1.php*, ottengo il seguente output

Java version=1.5.0_01
Java vendor=Sun Microsystems Inc.
OS=Linux 2.6.10-5-386 on i386
giovedì, aprile 14, 2005 at 3:59:53 PM Ora estiva dell'Europa centrale

informazioni tipiche di Java.

DOV'È JAVA ?

L'esempio che abbiamo appena visto lascia intuire la potenza del sistema, perché ci ha consentito di utilizzare classi messe a disposizione del framework J2SE 1.5.0 direttamente da PHP, ma non è ancora sufficientemente interessante. Un buon programmatore Java progetta delle classi, e vorrebbe poi riutilizzarle sia per le proprie applicazioni standalone sia per il web.
Consideriamo la seguente classe java

```
import java.util.LinkedList;
import java.util.List;

public class phptest{

    public List calcolaNumeriPrimi(int da, int a)
    {
       List risultato = new LinkedList();
       for(int i = da; i <= a; i++)
           if(numeroPrimo(i))
               risultato.add(new Integer(i));
       return risultato;
    }

    private boolean numeroPrimo(int n)
    {
        for(int i = 2; i <= Math.sqrt(n); i++)
           if(n % i == 0)
              return false;
        return true;
    }

    public static void main(String[] args)
    {
       phptest p = new phptest();
       if(args.length == 0)
       {
          System.out.println("errore - devi fornire due
                                                  numeri");
       } else
       {
          try
          {
```

Utilizza questo spazio per le tue annotazioni

```
            int arg2 = Integer.parseInt(args[1]);
            int arg1 = Integer.parseInt(args[0]);
            System.out.println(p.calcolaNumeriPrimi(
                                          arg1,arg2));
         } catch(NumberFormatException e)
         {
            System.out.println("errore - devi fornire
                                        due numeri");
         }
      }
   }
}
```

Si tratta dell'onnipresente classe per il calcolo dei numeri primi. Ce ne ha parlato anche Paolo Perrotta nel suo articolo sui *Thread* presente in questo stesso numero. In questo esempio l'abbiamo leggermente semplificata, ma nulla vieta che possiate aumentarne la complessità utilizzando i thread come nell'articolo in questione. Per i nostri scopi andrà benissimo questa versione.
Come vedete la classe in questione espone solo il metodo *calcolaNumeriPrimi(int da, int a)* che riceve in input due interi, a questo punto esegue un ciclo partendo dal primo numero e incrementando fino all'ultimo e per ogni ciclo verifica se il numero in corso è un numero primo o no. Se lo è lo aggiunge ad una Lista dichiarata di classe *List*. La verifica è affidata al metodo dichiarato private *numeroPrimo(int n)* che esegue i suoi calcoli e restituisce un valore booleano. Il *main* è strutturato in modo che debbano essere forniti due parametri all'applicazione, i due parametri vengono convertiti da stringhe a numeri e passati al metodo *calcolaNumeriPrimi(int da, int a)* infine viene stampata a video la lista restituita dal metodo. Non si tratta certo di una classe complessa. Ma è importante notare che si tratta di una classe java che a sua volta usa altre classi java in particolare i package *.util* ad esempio. Possiamo salvare il codice come *phptest.java* e compilare con *javac -Xlint phptest.java* e eseguirla con *java phptest*.
Ovviamente la classe si comporta esattamente come ci si aspetta, se non passiamo nessun parametro o dei parametri che ricalcano dei valori non numerici restituisce un errore, viceversa restituisce l'elenco dei numeri primi contenuti nell'intervallo. Ad esempio *java phptest 10 20* restituisce: *[11, 13, 17, 19]*

USIAMOLA DA PHP

Prima di tutto, è bene creare un jar delle nostre classi. Non è una regola, ma una buona norma di comportamento. In questo caso utilizziamo un solo file e una sola classe. Ma se ci muovessimo in ambiti più complessi avremmo molti file di classi e sarebbe indispensabile generare un *.rar* che li contiene.
Il comando è banale: *jar cvf phptest.jar phptest.class*, copiamo il *jar* appena generato in un qualche directory del nostro hard disk e passiamo al nostro file *php*.

```
<?
$here = trim(`pwd`);
java_set_library_path("$here/phptest.jar");
$numeri = new java("phptest");
$a=5;
$b=20;
foreach ($numeri->calcolaNumeriPrimi($a,$b)
                                        as $value)
{
    print($value)."<br>";
}
?>
```

In questo spezzone di codice ci sono diverse cose decisamente interessanti. Non consideriamo ovviamente la prima riga che ci restituisce semplicemente il percorso attuale. La seconda riga è già più utile, definisce il percorso della libreria che andremo a utilizzare: *java_set_library_path("$here/phptest.jar");* potremmo anche saltare questa istruzione, ma in tal caso la libreria dovrebbe essere raggiungibile nel *ClassPath* del sistema. Nella seconda riga si instanzia un oggetto di classe *phptest*. Da notare che è la classe che abbiamo definito in precedenza.
Infine, usiamo il metodo *calcolaNumeriPrimi ($a,$b)* direttamente da una sezione di *foreach*. In questa ultima istruzione è utile mostrare come il passaggio dei parametri avvenga tramite le variabili *$a* e *$b* che sono proprie di php.

UN PO' DI INTERAZIONE

Se siete un minimo esperti di PHP o di Web avrete già intuito che in questo paragrafo aggiungeremo alla nostra applicazione una form, per far sì che l'utente possa inserire da solo i due numeri che compongono il range da testare. Il codice è il seguente:

```
<form action="java4.php method=post>
 Valore 1: <input type=text name=a><br>
 Valore 2: <input type=text name=b><br>
  <input type=submit>
</form>
```

```
<?
$here = trim(`pwd`);

java_set_library_path("$here/phptest.jar");
$numeri = new java("phptest");
if ($_POST['a']!='' and $_POST['b']!='')
{
   foreach ($numeri->calcolaNumeriPrimi(
                        $_POST['a'],$_POST['b']) as $value)
   {
        print($value)."<br>";
   }
}
?>
```

Anche in questo caso il codice è estremamente comprensibile. Il passaggio dalla form a PHP avviene come di consueto tramite un *POST*, i parametri vengono passati all'oggetto Java tramite una normale interrogazione all'array *$_POST*. Un *if* verifica che le due variabili passate al metodo *calcolaNumeriPrimi* non siano delle stringhe vuote, onde evitare errori a runtime. Ovviamente si potrebbe migliorare il tutto effettuando i controlli sull'input, e qui c'è già da decidere se la validità sul formato dell'input deve essere effettuata direttamente da PHP o può essere delegata nella definizione del metodo *calcolaNumeriPrimi*. Nel primo caso saranno i programmatori PHP a intervenire, nel secondo i programmatori Java.

UN PO' DI COMMENTI

Spulciando nella documentazione relativa a *"Java Bridge"*, ovvero l'estensione che stiamo usando per realizzare queste tecniche, si trovano alcuni esempi interessanti. Alcuni sono addirittura relativo all'uso di J2EE ma, al di là di questi, ce n'è uno che contiene alcuni pezzi di codice che offrono spunti di riflessione. Si tratta di una miniapplicazione PHP che sfrutta una classe Java per creare dinamicamente un foglio Excel e salvarlo sull'Hard Disk.
Questo esempio in particolare ci sembra intelligente, perché in condizioni normali per creare un foglio di calcolo Excel via PHP dovremmo instanziare un oggetto COM e utilizzarne i metodi. La tecnica, per quanto documentata, non è esattamente un mostro di stabilità. Più di una volta si ottengono vistosi crash dell'applicazione, senza contare che per instanziare un oggetto COM il nostro PHP deve girare sotto Windows, e questo ovviamente sacrifica una parte importante dell'usare PHP, ovvero il suo essere multipiattaforma.

Viceversa integrando PHP con alcune classi java che manipolano direttamente i file Excel, tutto funziona a meraviglia anche sotto Linux e l'affidabilità è molto elevata rispetto a quella offerta dall'uso degli oggetti *com*.
Dando uno sguardo al codice troviamo:

```
java_set_library_path("$here/exceltest.jar;http://php
                              -java-bridge.sf.net/poi.jar");
$excel = new java("ExcelTest");
```

Il che è decisamente interessante. Notate infatti che la classe *"poi.jar"* viene invocata in modo remoto. Non è locale al computer dove risiede il web server, ma a sua volta è esposta su un web server su internet.
Il resto dovrebbe essere ormai abbastanza noto. Si potrebbe discutere su come internamente la classe in questione e gli esempi relativi realizzano il loro scopo, ma non è chiaramente l'oggetto di questo articolo.

CONCLUSIONI

Ci sono altre considerazioni da fare sull'uso di questa tecnica. In particolare vogliamo farvi notare che la classe phptest poteva anche essere implementata come runnable e questo avrebbe fatto sì che da php l'intero calcolo potesse essere eseguito in un Thread, con evidenti vantaggi per la velocità. Un esempio di come richiamare il thread da PHP potrebbe essere il seguente:

```
$Thread = new JavaClass("java.lang.Thread");
$Thread->start:
```

Ovviamente questo richiede degli accorgimenti. In questo stesso numero di ioProgrammo, Paolo Perrotta ci illustra appunto come si possano usare i thread in Java.
Chiaramente poiché PHP pur girando lato server proietta il suo Output su un browser, bisogna tenere conto del tempo che il browser impiega per renderizzare la pagina. Tuttavia in molti casi l'uso dei thread può velocizzare anche di molto un'applicazione.
Fino ad oggi questa tecnica non era stata applicata con frequenza, a causa di una certa debolezza dell'intero meccanismo, che poteva portare a qualche crash applicativo. La collaborazione fra Zend e Sun ha già molto migliorato questo aspetto rendendo il sistema decisamente più stabile. C'è ancora qualche passo da compiere prima di poter dire che tutto è perfetto, tuttavia ci sentiamo di affermare che la strada intrapresa è quella giusta.

http://php-java-bridge.sourceforge.net/

http://it2.php.net/manual/en/ref.java.php

Creare interfacce Java con XML

Un metodo molto interessante per separare la logica dell'interfaccia da quella dell'applicazione. La novità è che utilizzeremo un file esterno in formato XML per definire completamente la grafica di presentazione

Conoscenze richieste
Basi di Java
Basi di XML

Software
Java JSDK 1.4 o superiore, Thinlet

Impegno

Tempo di realizzazione

Thinlet è una libreria Java open source distribuita con licenza LGPL che permette di mantenere una netta separazione tra l'interfaccia e la logica applicativa grazie alla possibilità di leggere la definizione della GUI da file XML. Ogni componente, come bottoni e campi, dispone di una serie di proprietà quali visibilità e abilitazione settabili sia da XML che programmaticamente a runtime. Inoltre ogni componente genera una serie di eventi in funzione delle azioni compiute dall'utente su di esso. Ad ogni evento è possibile associare un metodo Java che definisce quale sarà l'azione corrispondente. Un'interfaccia grafica sviluppata con Thinlet è un albero di oggetti grafici. Ad esempio un "panel", che è l'elemento principale, può contenere una "label" per la visualizzazione di un testo generico, una *"textarea"* per l'immissione di un testo lungo ed una *"table"* per elencare dei risultati. La classe *Thinlet* è in grado di restituire per ogni elemento grafico dell'interfaccia un'istanza di Object da utilizzare come *"handle"*. L'impostazione delle proprietà dei componenti grafici è quindi delegata a *Thinlet,* che setta la proprietà dell'oggetto grafico a cui corrisponde l'handle passato come parametro. Come esempio, per impostare la proprietà *"enabled"* di un componente *"button"* si dovrà richiamare un metodo di *Thinlet* che imposta la proprietà del bottone passando l'handle dello stesso. Ad esempio:

```
Object buttonHandle = thinlet.create("button");
thinlet.setBoolean(buttonHandle, "enabled", true);
```

COME INIZIARE

Create una directory con nome *"ThinletDemo"* con tre sottodirectory: *"src"* dove troveranno posto i sorgenti Java, *"build"* utilizzata per memorizzare le classi compilate e *"lib"* dove sarà posizionato il file *jar* contentente la libreria *Thinlet.* Collegatevi al sito *www.thinlet.com*, dalla sezione *"download"* scaricate il file zip più aggiornato che al momento di scrivere l'articolo è *"thinlet-2004-03-07-zip"* e che contiene la documentazione ed il file *jar* della libreria. Scompattate lo zip e copiate il file *"thinlet.jar"* nella directory *"lib"* creata precedentemente. Oppure utilizzate direttamente il file che troverete nel cd allegato ad ioProgrammo.

DEFINIZIONE DELL'INTERFACCIA

Creiamo un nuovo file *"ToDoThinlet.xml"* nella cartella *"src"* che conterrà la definizione dell'interfaccia principale dell'applicazione. Tale file definisce un pannello nel quale il primo componente è la barra dei menu con l'unica voce *"Tasks"* che da accesso alla voce di menù per la creazione di un nuovo task. Alla voce di menu associamo l'event handler che aprirà la finestra di dialogo per l'inserimento di un nuovo task. Similmente il bottone provocherà l'invocazione del metodo *taskDone()* nel caso venga premuto.

```
<panel columns="1" gap="5" top="5" bottom="5"
       left="5" right="5" columns="1" weighty="1">
 <!-- barra dei menu -->
 <menubar weightx="1">
  <menu text="Tasks">
   <menuitem text="New event..." action=
                         "showNewEventDialog()"/>
  </menu>
 </menubar>
 <!-- tabella per l'elenco dei task da eseguire -->
 <table name="toDoTable" weighty="1">
 <header>
  <column text="Task"/>
 </header>
 </table>
```

```
<!-- tasto per l'evasione del task -->
<button text="done" weightx="1" action=
                                    "taskDone()"/>
</panel>
```

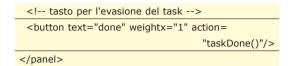
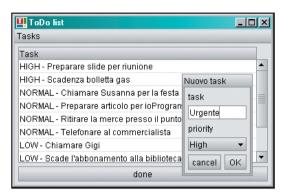



Fig. 1: ***Ecco come appare l'interfaccia della nostra applicazione***

Passiamo ora alla definizione del file XML che specifica l'aspetto grafico della finestra di dialogo per l'inserimento di un nuovo task. Qui troviamo due event handler, uno per la chiusura della finestra di dialogo, l'altro per la creazione del nuovo task. In questo caso l'elemento root non è *"panel"* ma *"dialog"* ad indicare che quello che si sta definendo andrà visualizzato come una finestra flottante all'interno dell'applicazione. L'attributo *"modal"* impostato a *"true"* specifica che non è possibile agire sull'applicazione sottostante sino a che la finestra di dialogo non è chiusa.

```
<dialog modal="true" columns="3" text=
                                    "Nuovo task">
   <label text="task" colspan="3"/>
   <textfield name="description" colspan="3"/>
   <label text="priority" colspan="3"/>
   <combobox name="priority" colspan="3">
         <choice text="High"/>
         <choice text="Normal"/>
         <choice text="Low"/>
   </combobox>
   <button text="cancel" colspan="2" action=
                        "closeNewEventDialog()"/>
   <button text="OK" action="newTask(
                  description.text,priority.selected)"/>
</dialog>
 La classe Task
```

La classe *Task* rappresenta l'oggetto principale trattato dall'applicazione. Un task ha la responsabilità di conoscere la propria descrizione e la propria priorità. Inoltre la classe implementa l'interfaccia *Comparable* in modo che le *Collection* di Task potranno essere facilmente ordinate in base alla priorità.

```
public class Task implements Comparable{
  public static final int HIGH=1;
  public static final int NORMAL=0;
```

```
  public static final int LOW=-1;
  private String description;
  private int priority;
  public Task(String description, int priority) {...}
  public String getDescription() {...}
  public int getPriority() {...}
  public String toString(){...}
  public int compareTo(Object o) {
    if(o instanceof Task){
      Task t = (Task)o;
      if(getPriority()!=t.getPriority())
        return t.getPriority() - getPriority();
      return getDescription().compareTo(
                              t.getDescription()); }
    throw new ClassCastException("not a task"); }
}
```

L'APPLICAZIONE

La classe importa il package thinlet ed estende la classe *Thinlet.* Sono definiti come attributi gli handle dei componenti grafici dell'interfaccia e l'array list che conterrà i task inseriti. Il costruttore si occupa di caricare i file XML che definiscono la finestra principale e la finestra di dialogo, memorizzando gli opportuni handle.

```
import [...];
import thinlet.Thinlet;
public class ToDoThinlet extends Thinlet{
    //task da visualizzare
    private ArrayList tasks;
    private Object toDoWindow;
    private Object toDoDialog;
    private Object toDoTable;
    private Object prioritiField;
    private Object descriptionField;
    public ToDoThinlet(){
        tasks = new ArrayList();
        //caricamento interfacce xml
        try {
            toDoWindow = parse("ToDoThinlet.xml");
            toDoDialog = parse("ToDoDialog.xml");
        } catch (IOException e) {
            throw new RuntimeException(e); }
        toDoTable = find(toDoWindow, "toDoTable");
        prioritiField = find(toDoDialog, "priority");
        descriptionField = find(toDoDialog, "description");
        //aggiunta della finestra di dialogo
        add(toDoWindow);
    }
```

Il primo metodo è il metodo *update()* che ha la responsabilità di creare la tabella con l'elenco di task inseriti. I metodi che creano e rimuovono *Task* invocheranno prima della loro conclusione *update()* in modo da ottenere l'aggiornamento dell'interfaccia.

EDITOR DI THINLET

Potete trovare ai seguenti indirizzi due editor per sviluppare interfacce con Thinlet. Entrambi permettono di creare l'interfaccia ed esportare poi l'XML equivalente.
Il primo, ThinG è un editor open source GPL mentre del secondo, Theodore, esistono una versione freeware ed una versione a pagamento.
http://thing.sourceforge.net/

http://www.carlsbadcubes.com/theodore/

Utilizza questo spazio per le tue annotazioni

EDITOR DI THINLET

Non è necessario indicare un DOCTYPE per gli XML di definizione dell'interfaccia. Tuttavia se avete un editor XML in grado di auto completare il codice, la definizione del DOCTYPE vi permette di ricevere preziosi aiuti dall'editor su quali elementi potete inserire all'interno di un elemento esistente.

COMPONENTI

Alcuni dei componenti grafici più interessanti previsti da Thinlet.

- **"tabbed" pane per la suddivisione logica di componenti grafici tramite pannelli identificati da linguette**
- **"spinbox" per l'impostazione di valori interi attraverso l'utilizzo di pulsanti per l'incremento e decremento del valore**
- **"progress" bar per mostrare l'avanzamento di un processo di elaborazione**
- **"slider" per l'impostazione rapida di valori numerici tramite cursore**
- **"split pane" per permettere all'utente di personalizzare l'interfaccia definendo lo spazio a disposizione di diversi insiemi di componenti**
- **"tree" per la visualizzazione di informazioni strutturate.**

La logica è quella di rimuovere tutte le righe eventualmente presenti nella tabella e per ogni task da evadere memorizzato nell'attributo di classe, aggiungere una riga con una cella contenente la descrizione sottoforma di stringa dell'oggetto ottenuta mediante *toString()*. La prima sezione ottiene l'elenco di tutte gli handle delle righe comprese nella tabella. Il metodo *getItems()* restituisce infatti tutti gli handle dei widget grafici contenuti in un componente di cui si fornisce l'handle. L'elemento *"table"* è costituito da molti componenti *"row"* a loro volta costituiti da componenti *"cell"*. Con un ciclo sugli handle ottenuti e attraverso il metodo *remove()*, vengono eliminate tutte le righe dalla tabella. A questo punto per ogni task presente nell'arrayList, viene creato a run time un oggetto *row* tramite il metodo *create*. Questo metodo accetta come parametro il nome di un qualsiasi componente grafico Thinlet, così come utilizzato nei file XML, e ne resituisce l'handle. Per ogni *row* viene aggiunta con la stessa logica una cell che contiene la descrizione del task corrente. Ogni volta sono anche resettati i controlli della finestra di dialogo.

```
public void update(){
    //svuotamento tabella
    Object[] rows = getItems(toDoTable);
    for (int i = 0; rows!=null && i < rows.length;
                                i++) {remove(rows[i]);}
    //aggiunta di task
    if (tasks != null) {
        Object row;
        Object objectCell;
        for (int i = 0; i < tasks.size(); i++) {
        //creazione della riga
            row = create("row");
            objectCell = create("cell");
            setString(objectCell, "text",
                                tasks.get(i).toString());
            add(row, objectCell);
            add(toDoTable, row);} }
    setString(this.descriptionField,"text","");
    setInteger(this.prioritiField, "selected", 1);
}
```

Veniamo ora agli *event handler*. In particolare è significativo l'handler specificato in corrispondenza del pulsante per l'inserimento di un nuovo task che invocherà il metodo *createNewTask* passando due argomenti. Il primo è la proprietà *text* del componente *description* che contiene la descrizione del task. Il secondo è il valore della proprietà selected del componente *priority* che è la priorità. Il metodo crea un nuovo task utilizzando la descrizione e la priorità specificate via interfaccia, aggiunge il task alla lista da visualizzare e riordina lo stesso con l'opportuno metodo statico *sort()* definito nella classe *java.util.Collections* che ci assicura che i task a priorità maggiore siano in cima alla lista. Viene poi richiamato il metodo *update()* per aggiornare la finestra.

```
 // Aggiunge un task alla lista mantenendola ordinata.
public void createNewTask(String description, int
                                prioritySelected){
    int prioriity=Task.LOW;
    if(prioritySelected==0){prioriity = Task.HIGH;}
    else if(prioritySelected==1){prioriity =
                                Task.NORMAL;}
    else if(prioritySelected==2){prioriity = Task.LOW;}
    Task task = new Task(description, prioriity);
    this.tasks.add(task);
    Collections.sort(this.tasks);
    closeNewEventDialog();
    update(); }
 // Marca come eseguito il task selezionato.
public void taskDone(){
    int index = getSelectedIndex(toDoTable);
    if(index!=-1) tasks.remove(index);
    update(); }
public void showNewEventDialog(){
    add(toDoDialog); }
public void closeNewEventDialog(){
    remove(toDoDialog);
}
```

Oltre al costruttore è necessario un metodo main che lanci l'applicazione. Thinlet mette a disposizione la classe di supporto FrameLauncher per fare ciò.

```
public static void main(String[] args) {
    ToDoThinlet toDo = new ToDoThinlet();
    FrameLauncher fl = new FrameLauncher(
                    "ToDo list", toDo, 500, 300);  }
}
```

COMPILARE E LANCIARE L'APPLICAZIONE

Portatevi nella directory *"src"* dei sorgenti e digitate il seguente comando avendo cura di sostituire *<home>* con la cartella del progetto.

```
javac -classpath <home>\lib\thinlet.jar -d ..\build
                                        *.java
```

Utilizzate il seguente comando per lanciare l'applicazione.

```
java -cp <home>\lib\thinlet.jar;<home>\build\
                                    ToDoThinlet
```

Utilizzate il menu per inserire nuovi task ed il pulsante sotto la tabella per evadere i task eseguiti.

Daniele de Michelis

Amici per la pelle thread e interfacce

Avete mai visto quelle interfacce grafiche che si "congelano" quando si preme un pulsante? Ecco come evitare questo comportamento maleducato grazie ai thread di Java

Uno dei problemi più comuni nei miei primi programmi Java, era quello delle "interfacce congelate". Queste interfacce si bloccavano ogni volta che il programma era impegnato a macinare dati. Se un programma faceva solo operazioni molto brevi, allora non c'era problema. Ma quando si trattava di fare lunghi calcoli, o magari di collegarsi ad un server remoto, il risultato era terribile: premevo un pulsante e tutto si bloccava, compreso l'aggiornamento dello schermo. Nei casi migliori, l'interfaccia era lenta come un bradipo; nei peggiori, finivo per chiedermi se il programma non fosse semplicemente andato in crash.

IL CALCOLO DI NUMERI PRIMI

Voglio scrivere un programma che permette di selezionare un valore iniziale e uno finale, e calcola tutti i numeri primi compresi tra i due valori. Eccone una prima versione, ancora incompleta:

```
import java.awt.*;
import java.util.*;
import javax.swing.*;
public class NumeriPrimi extends JPanel {
    private final JTextArea _output = new JTextArea();
                                  // Per visualizzare i risultati
    private final JSpinner _da = new JSpinner();
                                  // Il numero iniziale...
    private final JSpinner _a = new JSpinner();
                                  // ...e quello finale
    private final JButton _ok = new JButton("OK");
                                  // Per lanciare il calcolo
    public NumeriPrimi() {
        inizializzaGUI();
        // inizia il calcolo quando l'utente preme il pulsante
        _ok.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent ev) {
                calcolaNumeriPrimi();}});}
    private void calcolaNumeriPrimi() {
        _ok.setEnabled(false);   // Disabilita il pulsante
        _output.setText("");   // Cancella i risultati
                                                  precedenti
        // Calcola!
        List numeri = calcolaNumeriPrimi(((Integer)
                _da.getValue()).intValue(), ((Integer)_a
                                  .getValue()).intValue());

        // Stampa i risultati
        for(Iterator i = numeri.iterator(); i.hasNext();)
            _output.append(i.next() + "\n");
        _ok.setEnabled(true);   // Abilita il pulsante}
    private List calcolaNumeriPrimi(int da, int a) {
        return new LinkedList(); // FIXME: Per ora non
                                                  fa niente}
    // Piazza i componenti sul pannello
    private void inizializzaGUI() {
        setLayout(new BorderLayout());
        add(new JScrollPane(_output),
                                  BorderLayout.CENTER);
        JPanel input = new JPanel(new GridLayout(3, 2));
        input.add(new JLabel("Da:"));
        input.add(_da);
        input.add(new JLabel("A:"));
        input.add(_a);
        input.add(new JLabel(""));
        input.add(_ok);
        add(input, BorderLayout.SOUTH);}
    public static void main(String[] args) {
        JFrame f = new JFrame("Numeri primi");
                                  // Crea la finestra
        f.getContentPane().add(new NumeriPrimi());
                                  // Mettici dentro il pannello
        f.setSize(640, 480);  // Imposta le dimensioni
        f.setDefaultCloseOperation(JFrame
            .EXIT_ON_CLOSE); // Esci dal programma
                                  quando si chiude la finestra
        f.setVisible(true);  // Visualizza il bellissimo
                                                  risultato}
}
```

Fig. 1: *la finestra che mostra i numeri primi*

Il programma è tutto contenuto in un *JPanel*. Il *main()* crea un'istanza del *JPanel*, lo mette in una finestra (un *JFrame*) e lo visualizza. Non ho ancora scritto il codice che calcola i numeri primi; abbiate pazienza, mi sono concentrato sull'interfaccia. Il risultato è quello che si vede nella **Figura 1**: un paio di spinner (vedi box: *"Spinner"*) per impostare il numero iniziale e quello finale, un pulsante per calcolare i risultati e un'area di testo per mostrarli.
Il costruttore dispone tutti i componenti grafici sul pannello e collega il pulsante al metodo *calcolaNumeriPrimi()*. Questo metodo legge il valore iniziale e quello finale dagli spinner, lancia il calcolo e stampa i risultati nella *JTextArea*. Dato che sono un programmatore previdente, ho disabilitato il pulsante all'inizio del calcolo e l'ho abilitato nuovamente dopo la stampa dei risultati; così evito che l'utente possa premerlo più di una volta di seguito (ma sono stato troppo ottimista, come vedremo tra poco).

SEPARARE L'INTERFACCIA

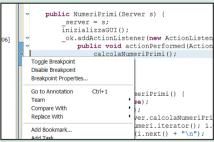
Il mio programma è lungo solo poche decine di righe, ma sta già prendendo una brutta piega. Non mi piace l'idea che il calcolo dei numeri finisca nella stessa classe che si occupa dell'interfaccia. Il codice delle interfacce tende sempre ad essere piuttosto complicato, e non voglio mescolarlo con la logica del programma. È sempre una buona idea separare le classi che si occupano della GUI da quelle che si occupano di fare i conti. Quindi scriverò una classe per contenere la logica, e farò in modo che *NumeriPrimi* deleghi i calcoli a un'istanza di questa classe. Darò a questa nuova classe il nome un po' pretenzioso di Server, per sottolineare che è lei a fornire i servizi alla GUI. Il Server potrebbe anche risiedere su un altro computer, magari su Internet (in questo caso diventerebbe forse un Web Service). Nel mio caso sarà semplicemente un oggetto locale:

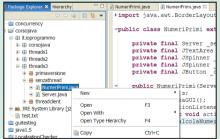
SPINNER

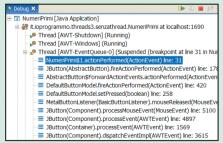
Non molti conoscono il termine "spinner", ma quasi tutti sanno cos'è. È una casella di testo speciale, fatta apposta per immettere numeri (di solito interi). L'utente può scrivere il numero direttamente nella casella, oppure può usare le due freccette a lato della casella per incrementare/decrementare il valore.

```
public class Server {
  public List calcolaNumeriPrimi(int da, int a) {
    List risultato = new LinkedList();
    for(int i = da; i <= a; i++)
      if(numeroPrimo(i))
        risultato.add(new Integer(i));
    return risultato;}
  private boolean numeroPrimo(int n) {
    for(int i = 2; i <= Math.sqrt(n); i++)
      if(n % i == 0)
        return false;
    return true;}
  public static void main(String[] args) {
    Server s = new Server();
    List primi = s.calcolaNumeriPrimi(3, 1000);
    for(Iterator i = primi.iterator(); i.hasNext();)
```

UNA SESSIONE DI DEBUG CON ECLIPSE

IMPOSTARE I BREAKPOINT

1 **Apri con Eclipse una qualsiasi versione del programma *NumeriPrimi*. Vai nel metodo *actionPerformed()*. Clicca con il tasto destro a sinistra della riga di codice. Apparirà un menu contestuale. Scegli *Toggle Breakpoint* per inserire un breakpoint, segnalato da un pallino azzurro: il programma si fermerà su questa riga durante il debugging**

MODALITÀ DI DEBUG

2 **Nel Package *Explorer* clicca con il tasto destro sul nome della classe e scegli *Debug -> Java Application*. Il programma *NumeriPrimi* partirà in modalità debug. Mentre il programma gira, clicca sul tasto *OK* per far eseguire il metodo *actionPerformed()*. Il programma si fermerà**

VEDIAMO I RISULTATI

3 **Eclipse dovrebbe avere aperto la "prospettiva" di debug. L'istruzione sulla quale ti sei appena fermato dovrebbe essere stata chiamato nell'ambito dell'*AWT EventQueue*, cioè la coda degli eventi del sistema grafico *AWT* di Java. Il thread principale dovrebbe essere già terminato: il *main()* del programma mostra la finestra e termina subito.**

```
System.out.println(i.next());}}
```

Il Server scopre se un numero è primo con un semplice algoritmo (vedi box: "*Sono contento di essere primo*"). I numeri primi, convertiti in *Integer*, finiscono nella lista dei risultati. Il *main()* contiene un test che calcola e stampa i numeri primi compresi tra 3 e 1000. L'ho lanciato. 3, 5, 7, 11, 13... Sì, direi che ci siamo. Ora la logica del programma è al suo posto.

SONO CONTENTO DI ESSERE PRIMO

Un numero primo è divisibile solo per *1* e per se stesso. Il modo più semplice per scoprire se un numero n è primo è provare a dividerlo per tutti i numeri da *2* a *n-1*:

```
private boolean numeroPrimo(int n)
{
  for(int i = 2; i < n; i++)
    if(n % i == 0)
    return false; //niente resto;
    // il numero è divisibile, quindi
    // non è primo
    return true; // non siamo riusciti
    // a dividere il numero,
    quindi è primo
}
```

In *NumeriPrimi* ho usato un algoritmo più efficiente: il ciclo delle divisioni va da *2* alla radice quadrata di *n*. Si può dimostrare che se non troviamo un divisore tra questi numeri, non lo troveremo nemmeno in seguito (la dimostrazione è facile, e i lettori con il pallino per la matematica possono provare a trovarla da soli).

```
private boolean numeroPrimo(int n)
{
  for(int i = 2; i <= Math.sqrt(n);
                                i++)
      // ...il resto non cambia
```

La classe *NumeriPrimi* potrebbe creare da sola un Server a cui delegare i calcoli, ma mi piace di più passargli un Server già fatto nel costruttore, in modo da poterlo eventualmente sostituire in seguito con un'altra implementazione (il famoso Web Service?). Ho evidenziato in grassetto le differenze principali rispetto alla versione precedente di *NumeriPrimi:*

```
public class NumeriPrimi extends JPanel...
   private final Server _server;
   public NumeriPrimi(Server s) {
      _server = s;
      <...>}
   private void calcolaNumeriPrimi() {
      _ok.setEnabled(false);
      _output.setText("");
      List numeri = _server.calcolaNumeriPrimi(((
         Integer)_da.getValue()).intValue(), ((Integer)
                        _a.getValue()).intValue());
      for(Iterator i = numeri.iterator(); i.hasNext();)
         _output.append(i.next() + "\n");
      _ok.setEnabled(true);}
   private void inizializzaGUI() {
      <...>}
   public static void main(String[] args) {
      JFrame f = new JFrame("Numeri primi");
      Server s = new Server();
      f.getContentPane().add(new NumeriPrimi(s));
      f.setSize(640, 480);
      f.setDefaultCloseOperation(
                          JFrame.EXIT_ON_CLOSE);
      f.setVisible(true);}}
```

Così mi piace di più. Ora che ho soddisfatto il mio senso estetico posso lanciare il programma, scrivere due numeri negli *"spinner"* e premere il pulsante. Ecco i risultati. Sembra tutto molto bello... finché non provo a fare qualche calcolo un po' più pesante. Ho chiesto al programma di calcolare i numeri primi tra dieci milioni e quindici milioni. I risultati non sono poi tantissimi, perché i numeri primi sono una vera rarità a queste quote. Ma come si può immaginare, il povero Server impiega un po' di tempo per fare tante divisioni: il mio Pentium a 3GHz richiede una quarantina di secondi per arrivare alla stampa dei risultati. La cosa seccante è che durante questo tempo, il programma sembra morto. Le barre di scroll e gli spinner non funzionano. Se riduco le dimensioni della finestra e poi la ingrandisco di nuovo, il sistema non aggiorna nemmeno la visualizzazione sullo schermo: il risultato è il bellissimo macchione grigio che si può ammirare nella **Figura 2**. Non è finita: la casella dei risultati non viene svuotata prima del calcolo, anche se il programma lo chiede esplicitamente *(_output .setText(""))*. Evidentemente il sistema grafico di Java non fa in tempo ad aggiornare lo schermo prima che il lungo calcolo dei risultati lo blocchi. E se avessi voluto dare all'utente una barra di avanzamento, o qualche altro strumento per indicare il tempo restante per finire il calcolo? Sarebbe stato tutto inutile: senza il "refresh" del video, la mia barra sarebbe apparsa solo al termine dell'operazione. Brutto. Inaccettabile. Qui bisogna correre ai ripari.

Fig. 2: l-interfaccia non viene aggiornata durante l'elaborazione dei dati

Il box "*La coda degli eventi*" spiega perché l'interfaccia si blocca, e suggerisce una soluzione. Chi non sa cosa sia la coda degli eventi di Java, dovrebbe leggere questo box, prima di continuare.

USIAMO I THREAD

Per evitare che l'interfaccia si fermi durante le operazioni lente, devo fare in modo che il calcolo avvenga in un thread separato. In questo modo il controllo tornerà alla coda degli eventi di Java appena possibile. Così la coda degli eventi potrà occupare il proprio tempo in cose utili come aggiornare lo scher-

mo, anziché restare in attesa sulla soglia per quaranta interminabili secondi.

```
public class NumeriPrimi extends JPanel...
    public NumeriPrimi(Server s) {
        _server = s;
        inizializzaGUI();
        _ok.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent ev) {
                lanciaCalcoloInNuovoThread();}});}
    private void lanciaCalcoloInNuovoThread() {
        Runnable calcolo = new Runnable() {
            public void run() {
                calcolaNumeriPrimi();}};
        Thread t = new Thread(calcolo);
        t.start(); }
```

Anziché chiamare direttamente il metodo *calcolaNumeriPrimi()*, l'azione legata al pulsante chiama *lanciaCalcoloInNuovoThread()*. Questo metodo definisce (con una classe anonima) un nuovo *Runnable,* il cui metodo *run()* chiama a sua volta *calcolaNumeriPrimi()*. Il *Runnable* viene usato per costruire un *Thread,* che parte subito. Il metodo *run()* girerà in un thread parallelo, senza costringere la coda degli eventi ad aspettare la fine del calcolo. L'ho provato, e funziona: l'interfaccia non si congela più. Sembra che quest'ultima versione del programma vada bene. Ma quando si tratta dei thread, le cose sembrano spesso più semplici di quello che sono.

SINCRONIZZIAMO

Ho fatto qualche prova per verificare se è possibile premere il pulsante più di una volta, lanciando più calcoli in parallelo. Con l'aiuto di qualche stampa sullo schermo, ho scoperto che un utente dal dito lesto riesce facilmente a premere il tasto due volte di seguito. Il problema è che lanciare un thread è un'operazione lenta; quindi passa un po' di tempo tra il momento in cui il controllo torna alla coda degli eventi e il momento in cui il pulsante viene disabilitato in *calcolaNumeriPrimi()*. Durante questo breve intervallo, è possibile premere una seconda volta il pulsante. Per risolvere il problema ho sincronizzato il metodo *lanciaCalcoloInNuovoThread()*. Ora il metodo non può essere eseguito in parallelo da più thread sulla stessa istanza del programma. Un'alternativa è quella di disabilitare il pulsante appena possibile, cioè nel thread degli eventi anziché in quello dei calcoli. Ecco una versione del programma che fa entrambe le cose:

```
private synchronized void lanciaCalcoloInNuovoThread() {
        _ok.setEnabled(false);
        <...>
```

Le sincronizzazioni sono bestiole pericolose, che richiedono molta attenzione. Vale la pena di dare un'ultima occhiata al programma, per essere sicuri di non aver dimenticato qualche risorsa condivisa che potrebbe causare problemi.
In questa circostanza devo essere certo che nessun campo di *NumeriPrimi* possa essere letto/modificato in parallelo da più thread. Ad esempio, due thread potrebbero cercare di stampare contemporaneamente sulla *JTextArea.* Ora che *lanciaCalcoloInNuovoThread()* è sincronizzato, non mi vengono in mente situazioni nelle quali potrebbe succedere una cosa simile. Direi che posso essere abbastanza sicuro. Quanto alla classe *Server,* non può causare problemi di sincronizzazione, perché non ha campi; quindi non ha uno stato che possa essere condiviso da più thread. Anche se molti client usassero questa classe in parallelo, non ci sarebbero problemi. Ma se avessi molti client cercherei probabilmente di rendere il sistema un po' più veloce, e questo potrebbe significare aggiungere dei campi al Server.
Ad esempio potrei conservare una cache dei numeri primi già calcolati per determinati intervalli. Se lo facessi, l'accesso a questa cache dovrebbe essere sincronizzato. In caso contrario, un client potrebbe cercare di modificarne i valori mentre un altro client cerca di leggerli da un thread diverso.

Paolo Perrotta

LA CODA DEGLI EVENTI

Per quale motivo l'interfaccia grafica si "congela" quando parte un'operazione molto lenta? In un programma Java sono attivi, in ogni momento, diversi thread. Uno molto famoso è il thread principale, cioè il thread all'interno del quale viene chiamato il metodo *main().* Nel caso di un programma grafico come NumeriPrimi, però, il *main()* termina quasi subito. Eppure il programma non termina, perché restano in vita altri thread importanti. Uno di questi è *la coda degli eventi.* La cosa degli eventi si occupa di gestire tutti gli eventi del sistema grafico. Se l'utente preme un pulsante, questo evento finisce nella *coda degli eventi.* DI conseguenza la coda degli eventi chiama il metodo *ActionListener.actionPerformed()* di tutti i *"listener"* collegati al pulsante. Se non ci fosse la coda degli eventi, il sistema non si accorgerebbe che l'utente ha premuto il tasto - a meno che il programma non fosse fermo e concentrato sulla lettura di quel particolare tasto. Il metodo *actionPerformed()* non viene chiamato dal thread principale, ma dal thread degli eventi. Il thread degli eventi si ferma fino a quando il *actionPerformed()* non è terminato – nel nostro caso, fino a quando il metodo *calcolaNumeriPrimi()* non ha finito di fare conti. Nel frattempo, tutti gli eventi nel sistema sono sospesi. È inutile cliccare su un altro pulsante: nessuno sarà lì a ricevere questo evento. E purtroppo anche gli aggiornamenti dello schermo sono eventi, quindi anch'essi resteranno bloccati in attesa. Questo è il motivo per cui, in risposta ad un evento della GUI, dovremmo sempre cercare di restituire il controllo il prima possibile. Questo purtroppo non è sempre facile. In questo articolo risolviamo il problema con un nuovo thread, parallelo sia al thread principale (se è ancora in vita) che alla coda degli eventi. Questo thread si occupa del calcolo parallelo, e il controllo torna subito al sistema grafico.

Mono, un ponte fra Windows e Linux

Per quanto possa sembrare strano, è possibile programmare utilizzando .NET e C# anche in ambiente Linux. È Mono a consentire tutto questo, vediamo come funziona e quali vantaggi porta

Mono è un'implementazione open source della piattaforma .NET. Essa include gli strumenti, le librerie, e il compilatore richiesti per costruire software su varie piattaforme quali: Linux, Windows e MacOS. Abbraccia, inoltre, diverse architetture, dalla x86 ai computer s390. In quest'articolo vedremo come installare Mono, sia su Linux sia su Windows, ed esamineremo alcuni degli strumenti che esso ci fornisce. Infine utilizzeremo C# per scrivere due esempi di tipo Console e uno GUI. Quest'ultimo con l'ausilio di Gtk#, che è la libreria preferita per la costruzione di applicazioni GUI, usando Mono.

PERCHÉ MONO?

Gli sviluppatori Linux saranno sicuramente interessati a Mono, ma un programmatore Windows potrebbe chiedersi: "Perché dovrei interessarmi a Mono se per Windows c'è già .NET Framework?" La risposta è semplice.
A differenza di .NET di Microsoft, Mono è multipiattaforma. Questo fa sì che un programma compilato con Mono "giri" indifferentemente su qualsiasi piattaforma su cui Mono è installabile. Eseguire un'applicazione sviluppata sotto Linux, su un altro sistema operativo che supporta Mono, sarà tanto semplice quanto copiare gli assembly relativi ad essa sul sistema target. Capiamo, quindi, che nessuno sviluppatore, .NET e non, dovrebbe fare a meno di interessarsi almeno minimamente a Mono.

INSTALLAZIONE SU LINUX

Sicuramente il modo più semplice per installare Mono su Linux è usare i package ufficiali. Gli utenti di sistemi RPM-based (Red Hat, SuSE, Fedora Core) possono scaricare gli RPM adatti alla loro distribuzione di Linux. L'URL di riferimento è *http://www.mono-project.com/downloads/*.
Inoltre altre distribuzioni di Linux offrono supporto per Mono tramite canali dedicati.
Per installare Mono attraverso i package:

1. Puntare il browser all'URL sopra citato
2. Cliccare su Packages accanto alla propria distribuzione
3. Cliccare sul package che inizia per mono-core-1.0.5 nella sezione in alto a sinistra dal titolo Mono Core Runtime and C# compiler
4. Aprire il file e procedere all'installazione

A questo punto il package Mono core dovrebbe essere installato. In ogni caso, anche se non sono disponibili i package per la propria distribuzione, è sempre possibile installare Mono dai sorgenti.

INSTALLAZIONE SU WINDOWS

L'installazione su Windows sarà molto semplice. Puntiamo il browser alla pagina *http://www.go-mono.com* e andiamo all'area download. Scarichiamo l'installer e lanciamolo facendo il classico doppio click su esso. Questo installerà Mono in *C:\Programmi\Mono-1.0.5* o nella directory da noi specificata in fase di installazione.
Sotto la cartella bin troviamo i file batch che "wrappano" gli eseguibili che useremo per compilare ed eseguire i programmi e fare altre operazioni. A questo punto conviene aggiungere il percorso *C:\Programmi\Mono-1.0.5\bin* alla path.
Questo ci permetterà di usare i comandi di Mono da qualsiasi directory ci troviamo.

REQUISITI

Conoscenze richieste
Basi di C#

Software
Mono 1.0.5, Windows 2000 o superiore o Linux

Impegno
Tempo di realizzazione

SCRIVIAMO UN PO' DI CODICE

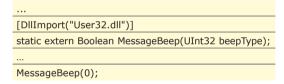
Facciamo un indovinello. Quale sarà il primo programma che scriveremo? Indovinato! Proprio "Hello World". Aprite il vostro editor di testi preferito e digitate il seguente codice:

```
using System;
public class HelloWorld {
    static void Main() {
        System.Console.WriteLine("Hello World");}
}
```

Salviamo il file col nome *Hello.cs*. A questo punto, per compilarlo, basta eseguire il comando *mcs Hello.cs*. Se tutto è andato a buon fine, dovrebbe comparire il messaggio *"Compilation Succeeded"* ed il risultato sarà il file *Hello.exe* che si troverà nella stessa directory dalla quale abbiamo lanciato il comando. Se invece vi dà qualche errore, assicuratevi di aver aggiunto alla PATH i percorsi indicati nel paragrafo *"Installazione su Windows"* o, se proprio non volete farlo, di essere sotto la directory bin di mono. A questo punto non ci resta che lanciare la nostra prima "applicazione" compilata con Mono. Per farlo, eseguite il seguente comando: *mono Hello.exe*.
Il risultato è la stringa *Hello World* visualizzata sulla Console. Quello che vogliamo mettere in evidenza, arrivati a questo punto, è una caratteristica molto interessante. Se siete su piattaforma Windows e avete .NET Framework installato, scrivete *Hello.exe* e premete invio. L'output sarà analogo a quello precedente e cioè Hello World. Sarebbe lecito chiedersi a questo punto: *"Ok e qual è la caratteristica interessante di ciò, dato che abbiamo ottenuto lo stesso Hello World dell'istruzione precedente?"*. La caratteristica molto interessante sta nel fatto che il *Common Intermediate Language (CIL)*, che abbiamo ottenuto compilando il programma con Mono, è stato eseguito senza batter ciglio dal CLR del .NET Framework di Microsoft! Ma com'è possibile ciò? La spiegazione sta nel fatto che il CLR è l'implementazione di Microsoft delle specifiche del *Common Language Infrastructure (CLI)*.
Quest'ultimo è uno standard internazionale per creare ambienti di sviluppo ed esecuzione in cui i linguaggi e le librerie lavorano assieme senza problemi. Il fatto che sia uno standard ha reso possibile l'implementazione dello stesso da parte di Mono. Questo fa sì che un programma scritto e compilato con Mono possa essere eseguito dal CLR di Microsoft. È vero anche il contrario, cioè se compiliamo l'esempio precedente col compilatore di Microsoft, questo potrà essere eseguito facendo uso di Mono. Credo che ora stia iniziando a venire fuori l'importanza di quest'ultimo. Sfortunatamente, "per ora", non è tutto così bello come sembra. Il problema è che non tutto il codice compilato con Mono può essere eseguito usando il CLR di Microsoft e non tutto il codice compilato col compilatore della casa di *Redmond* può essere eseguito usando Mono. La spiegazione a ciò sta nella definizione di applicazione 100% .NET.

APPLICAZIONI 100% .NET

Un'applicazione è 100% .NET se fa uso esclusivamente di API definite sotto il namespace *System* e non fa chiamate di tipo *P/Invoke (Platform Invoke)*. Una chiamata di tipo *P/Invoke* può essere, ad esempio, una chiamata ad un'API di Windows (e quindi *unmanaged code*) da codice managed scritto in C#. Ad esempio, per chiamare l'API *MessageBeep* di Windows potremmo usare il seguente codice all'interno del nostro programma C#:

```
...
[DllImport("User32.dll")]
static extern Boolean MessageBeep(UInt32 beepType);
...
MessageBeep(0);
```

In questo caso, il problema risiede nel fatto che abbiamo legato la nostra applicazione ad una piattaforma specifica (Windows) e ciò fa cadere il motivo per il quale Mono è nato e cioè l'essere multipiattaforma. A parte questo, c'è un altro problema. Mono non è "ancora" un'implementazione completa di .NET. Diciamo "ancora" perché l'obiettivo primario del progetto è di implementare tutte le API del framework .NET. Ad esempio, una buona parte delle classi presenti in *System.Windows.Forms*, non è ancora stata implementata. In **Figura 1** è visibile lo stato in cui si trova l'implementazione di *System.Windows .Forms*. Come si può vedere una buona parte del namespace non è ancora

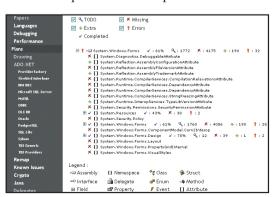



Fig. 1: **Stato in cui si trova l'implementazione di System.Windows.Forms**

Sito del progetto:
http://www.mono-project.com/about/index.html

Stato delle librerie di classi di Mono:
http://www.go-mono.com/class-status.html

Pagina relativa al progetto Gtk#:
http://gtk-sharp.sourceforge.net/

Specifiche per C#:
http://www.ecma.ch/ecma1/STAND/ecma-334.htm

Specifiche per il CLI:
http://www.ecma.ch/ecma1/STAND/ecma-335.htm

• **MONO: A DEVELOPER'S NOTEBOOK**
Niel M. Bornstein
Edd Dumbill
(O'Reilly)

MODIFICA DELLA PATH SU WINDOWS

Per impostare la PATH su Windows 2000 o XP bisogna eseguire i seguenti passi: selezionare *Impostazioni* dal menu *start,* ed aprire il *Pannello di Controllo;* quindi selezionare *Sistema.* Selezionare il tab *Avanzate.* Selezionare *Variabili d'ambiente* e cercare *Path* tra le *Variabili di Sistema.* Aggiungere il percorso *;C:\Programmi \Mono-1.0.5\bin* all'estremità destra della variabile *Path.* Premere *OK.*

Regole per realizzare applicazioni .NET multipiattaforma

- **Preferire l'uso di Gtk# a System.Windows.Forms**
- **Evitare il più possibile l'uso di *P/Invoke* su librerie native.**
- **Non leggere le chiavi di registro, attraverso *Microsoft.Win32.Registry,* che sono supportate da Windows.**
- **Non usare percosi assoluti.**

Ad esempio:

```
// SBAGLIATO
string percorso =
        directory + "/" +
              filename;
// GIUSTO
string percorso =
   System.IO.Path.Combine
     (directory, filename);
```

implementata. Ovviamente le librerie di classi già implementate in Mono, non sono tali da permetterci di scrivere solo il famigerato Hello World. A riprova di ciò scriviamo un programmino che estragga da un URL tutti i link presenti all'interno dello stesso e li scriva in un file di testo.
Per fare ciò useremo i seguenti namespace: *System, System.IO, System.Text.RegularExpressions* e *System.Net.*
Il listato è il seguente:

```
using System;
using System.IO;
using System.Text.RegularExpressions;
using System.Net;
public class LinksSniffer
{ static void Main(string[] args)
  { if(args.Length != 1) {
      Console.WriteLine("Sintassi: LinksSniffer URL");
       return;}
    StreamReader reader = null;
    StreamWriter writer = null;
    try {
      writer = new StreamWriter("links.txt", false);
      WebRequest request =
                         WebRequest.Create(args[0]);
      WebResponse response = request.GetResponse();
      reader = new StreamReader(
                       response.GetResponseStream());
     Regex regex = new Regex("<a.*?href\\s*=\\s*\
         "([^\"]*)\"", RegexOptions.IgnoreCase);
     string line = "";
     while((line = reader.ReadLine()) != null) {
        MatchCollection matches = regex.Matches(line);
        foreach(Match singleMatch in matches)
        writer.WriteLine(singleMatch.Groups[1]);}}
    catch (Exception e) {
      Console.WriteLine(e.Message);}
    finally {
       if(writer != null)
          writer.Close();
       if(reader != null)
          reader.Close();}}}
```

Salviamo il programma col nome *LinksSniffer.cs* e compiliamolo con: *mcs LinksSniffer.cs.*
Questa classe non fa altro che prendere l'URL che passiamo da riga di comando ed effettuare il parsing del codice html tramite l'uso di un'espressione regolare. Ogni occorrenza che rispetta il pattern della *Regex*, è scritta in un file di testo di nome *links.txt.* Un esempio d'uso è: mono *LinksSniffer.exe http://www.mono-project.com /about/index.html.* Chiaramente l'utilità di tale classe è ampiamente opinabile.
L'unico scopo in questo contesto era dimostrare che Mono ha già implementato abbastanza classi delle librerie .NET. Il progetto è comunque in continua espansione e lo scopo principale è quello di implementare .NET al 100% . A parte questo, sono stati e saranno implementati, altri toolkit che faciliteranno la vita di noi sviluppatori. Tra questi emerge *Gtk#* il quale permetterà lo sviluppo di interfacce grafiche utente (*GUI*) con l'importante caratteristica di essere multipiattaforma.

APPLICAZIONI GUI CON GTK#

Fig. 2: ***Hello World GUI con Gtk# su Windows***

Prima di concludere la nostra panoramica su Mono non potevamo fare a meno di costruire un semplicissimo esempio di applicazione GUI usando *Gtk#*. *Gtk#* è il "wrapper" Mono per Gtk+. Quest'ultimo è ampiamente utilizzato per sviluppare interfacce grafiche, soprattutto in ambiente Linux.
A tal proposito, arrivati a questo punto, gli utenti Linux dovranno scaricare ed installare il package relativo a Gtk# dalla pagina *http://www.mono-project.com/downloads/* seguendo una procedura analoga a quella dell'installazione del package *mono-core-1.0.5* vista nel paragrafo *Installazione su Linux* di quest'articolo. Gli utenti Windows invece non dovranno preoccuparsi di ciò poiché Gtk# è già incluso nell'installer. Senza scendere troppo nei dettagli, costruiamo il nostro primo semplice programma usando Gtk#.
Esso sarà composto solamente da una finestra e un bottone. Ogni volta che sarà cliccato il bottone apparirà la stringa ioProgrammo sulla Console. Il risultato finale sarà quello visibile in **Figura 2**.
Il listato relativo ad esso è il seguente:

```
using Gtk;
using GtkSharp;
using System;
public class Hello {
   static void Main() {
      Application.Init ();
      Window window = new Window ("Hello World");
      Button btn = new Button ("Cliccami");
      btn.Clicked += new EventHandler (HelloEvent);
      window.DeleteEvent += new DeleteEventHandler
                              (WindowDeleteEvent);
      window.Add (btn);
      window.ShowAll ();
      Application.Run ();}
```

```
static void WindowDeleteEvent (object obj,
                    DeleteEventArgs args) {
Application.Quit ();}
static void HelloEvent (object obj, EventArgs args) {
Console.WriteLine("ioProgrammo");}}
```

Per compilare questo programma usare il seguente comando:

sotto Windows:

```
mcs hellogtk.cs –lib:"C:\Programmi\Mono-1.0.5
        \lib\mono\gtk-sharp" –r:gtk-sharp.dll
```

sotto Linux:

```
mcs hellogtk.cs –r:gtk-sharp –lib:/usr/lib/mono
                                    /gtk-sharp
```

Ovviamente se avete installato Mono su una directory diversa, dovrete adattare il percorso in *–lib* di conseguenza. Nel nostro caso, *hellogtk.cs* è il nome che abbiamo dato al file e *gtk-sharp.dll* è la libreria relativa a Gtk#. L'opzione *–r* indica che stiamo referenziando quella libreria, mentre *–lib* indica dove si trova. Per eseguirlo basta editare: mono *hellogtk .exe*. È possibile notare che, come ogni toolkit per la costruzione di interfacce grafiche, anche Gtk# è guidato dagli eventi. Questo significa che la GUI "dormirà" in *Application.Run()* finché non si verificherà un evento. In tal caso il controllo è passato al metodo associato a quell'evento particolare. Infine c'è una cosa da notare. L'assegnazione dei gestori degli eventi l'abbiamo fatta utilizzando la notazione estesa per enfatizzare l'uso dei delegati. Nella pratica potrebbe preferirsi la notazione breve.
Quindi possiamo scrivere:

```
window.DeleteEvent += WindowDeleteEvent;
btn.Clicked += HelloEvent;
```

al posto di:

```
window.DeleteEvent += new DeleteEventHandler
                    (WindowDeleteEvent);
btn.Clicked += new EventHandler (HelloEvent);
```

raggiungendo lo stesso identico risultato.

CONCLUSIONI

In quest'articolo abbiamo visto alcune delle caratteristiche di Mono. Il progetto è ancora molto giovane (la prima release è avvenuta a fine giugno '04) ma non per questo da sottovalutare. Basti pensare al fatto che Mono supporta anche ASP .NET, ADO.NET e MonoBASIC (Visual Basic .NET) anche se ancora non completamente. Diamogli il tempo di lavorare e potremo finalmente scrivere anche la nostra Web Application in ASP .NET e vederla "girare" indifferentemente, sia su server Windows sia su server Linux.

Alessandro Lacava

INSTALLAZIONE SU LINUX DAI SORGENTI

Per eseguire l'installazione dai sorgenti:

1 Scaricare i sorgenti andando all'URL http://www.mono-project.com/downloads/.
2 Loggarsi come utente root
3 Decomprimere il file scaricato usando il comando tar *-xvzf mono-1.0.5.tar.gz*
4 Andare sotto la directory *mono-1.0.5*
5 Eseguire il comando *./configure*
6 Per compilare i sorgenti eseguire il comando *make*
7 Per eseguire l'installazione vera e propria eseguire *make install*

HELLO WORLD GUI CON GTK#

IMPORTIAMO I NAMESPACE

```
using Gtk;
using GtkSharp;
using System;
```

1 **Questi sono i namespace che contengono le classi che ci interessano per la costruzione della nostra GUI tra le quali Window e Button. Sono sufficienti per iniziare.**

CREIAMO GLI OGGETTI DELLA NOSTRA GUI

```
Application.Init ();
Window window = new Window ("Hello
World");
Button btn = new Button ("Cliccami");
```

2 **Init fa sì che la libreria Gtk# sia inizializzata. Di seguito viene creata la finestra principale che avrà come titolo Hello World e il bottone che avrà come testo la stringa Cliccami**

GESTIAMO GLI EVENTI

```
btn.Clicked += new EventHandler
                         (HelloEvent);
window.DeleteEvent += new
  DeleteEventHandler (WindowDeleteEvent);
```

3 **Queste due righe di codice non fanno altro che, associare all'evento relativo alla pressione del bottone e a quello relativo alla chiusura dell'applicazione, i corrispondenti metodi.**

VISUALIZZIAMO LA FINESTRA

```
window.Add (btn);
window.ShowAll ();
Application.Run ();
```

4 **Prima di visualizzare la finestra viene aggiunto ad essa il bottone creato in precedenza. Run fa sì che la GUI venga mostrata e, cosa più importante, la mantiene aperta, aspettando che si verifichino eventi, finché viene chiamato Quit.**

GESTIAMO LA PRESSIONE DEL BOTTONE

```
static void HelloEvent (object obj,
                        EventArgs args)
{
  Console.WriteLine("ioProgrammo");
}
```

5 **Come abbiamo visto al passo 3, questo è il metodo che è stato associato all'evento relativo alla pressione del bottone. Tale metodo non fa altro che visualizzare a Console la stringa ioProgrammo.**

GESTIAMO LA CHIUSURA DELLA FINESTRA

```
static void WindowDeleteEvent (object obj,
                        DeleteEventArgs args)
{
  Application.Quit ();
}
```

6 **Analogamente al passo 5 questo metodo viene chiamato quando di verifica un evento. In questo caso tale metodo è associato all'evento relativo alla chiusura della finestra. Il suo scopo è semplicemente quello di chiudere la GUI.**

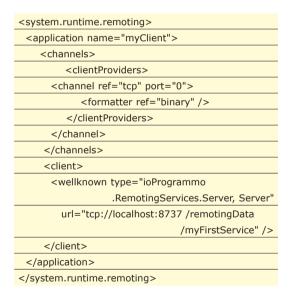
.NET Remoting: software che comunicano

Vedremo come creare host e invocare oggetti remoti da codice senza l'uso di file di configurazione. Impareremo come accedere al server usando un'interfaccia ed effettuare una chiamata bidirezionale

Chi segue costantemente ioProgrammo, ricorderà che nel numero precedente abbiamo illustrato le basi di .NET remoting, ovvero la tecnologia che consente a due programmi residenti su macchine diverse di comunicare fra loro. Il fine di questa tecnologia è quello di consentire la creazione di applicazioni distribuite, tali che piccoli spezzoni di codice, specializzati nella risoluzione di un solo singolo problema possano essere dislocati ovunque all'interno di una rete aziendale o nel mondo. Le varie parti del software comunicando fra loro contribuiranno a formare un'unica grande applicazione che usufruisce di un'elevata capacità computazionale derivata dall'usare molte macchine, le quali hanno un carico ridotto di compiti. Ci sono altri vantaggi evidenti nell'uso di questa tecnica, ne abbiamo diffusamente parlato nello scorso numero. Nell'illustrare questa tecnologia avevamo mostrato come le direttive relative a porta su cui un servizio si mette in ascolto, protocollo di trasporto e le altre direttive relative alla configurazione client /server potessero risiedere su un file .config, separato dal codice dell'applicazione. In questo articolo illustreremo la tecnica inversa, ovvero configureremo l'applicazione client e l'applicazione host inserendo le direttive direttamente nel codice.

USARE REMOTING SENZA FILE DI CONFIGURAZIONE

L'uso del file di configurazione per impostare le informazioni di remoting è estremamente comodo perché consente, con una sola istruzione, di fruire in modo trasparente di questa tecnologia senza minimamente modificare il proprio codice. È però vero che tale modello può risultare in taluni casi un po' rigido. Può rendere complicata l'implementazione di politiche di configurazione più sofisticate o legate alla logica applicativa, al punto che una possibilità nata come semplificazione all'uso della tecnologia può trasformarsi in una limitazione. Ed ecco la soluzione: configurare remoting da codice e senza l'uso di file .config. Partiamo innanzitutto dal client. Prima di continuare, è utile definire il client come l'applicazione che usufruisce di un servizio remoto, e l'host come l'applicazione che espone il servizio. Diremo formatter il modello secondo cui i dati vengono scambiati, diremo channel il protocollo di comunicazione utilizzato.
Supponiamo di voler ricalcare via codice il seguente file di configurazione esterno:

```
<system.runtime.remoting>
  <application name="myClient">
      <channels>
           <clientProviders>
        <channel ref="tcp" port="0">
              <formatter ref="binary" />
           </clientProviders>
        </channel>
      </channels>
      <client>
        <wellknown type="ioProgrammo
                 .RemotingServices.Server, Server"
          url="tcp://localhost:8737 /remotingData
                              /myFirstService" />
      </client>
  </application>
</system.runtime.remoting>
```

Gli elementi che questo file configura sono fondamentalmente tre: il formatter usato, il channel usato e il tipo da richiamare via remoting. Tenteremo di ricalcare la stessa configurazione, all'interno del codice di un'ipotetica applicazione. Innanzitutto dobbiamo aggiungere nel progetto client un riferimento ad un assembly di sistema registrato nella GAC ed in particolare System.Runtime.Remoting.Dll. A questo punto possiamo procedere con la dichiarazione delle using corrispondenti:

```
using System.Runtime.Remoting;
using System.Runtime.Remoting.Channels;
using System.Runtime.Remoting.Channels.Tcp;
```

Cominciamo con la definizione del formatter che, nel nostro caso, è il classico BinaryFormatter:

```
BinaryClientFormatterSinkProvider formatterProvider
            = new BinaryClientFormatterSinkProvider();
```

Tale oggetto può essere associato soltanto al channel, come si evince anche dalla struttura xml del file di configurazione, e dunque passiamo a definire quest'ultimo:

```
if ( ChannelServices.GetChannel("tcp") == null ) {
    IDictionary props = new Hashtable();
    props["name"] = "tcp";
    props["port"] = 0;
    TcpChannel channel = new TcpChannel(
                        props, formatterProvider, null);
    ChannelServices.RegisterChannel(channel);
}
```

Abbiamo quindi semplicemente istanziato un channel di tipo TcpChannel passando al costruttore le proprietà name e port. Quest'ultima potrebbe sembrare inutile nel client, ma è necessaria quando si vuole instaurare una comunicazione bidirezionale, come osserveremo nel proseguo. Al costruttore del channel passiamo anche il formatter legandoli così in modo indissolubile. A questo punto semplicemente registriamo il channel nel sottosistema di remoting della nostra applicazione .NET. Per precauzione verifichiamo prima che il channel non sia stato già precedentemente registrato in modo da evitare errori a runtime. Infine possiamo effettuare la chiamata al nostro oggetto ioProgrammo.RemotingServices.Server. Quando facevamo uso nel nostro file di configurazione, l'istanziazione era una banale:

```
Server server = new Server();
```

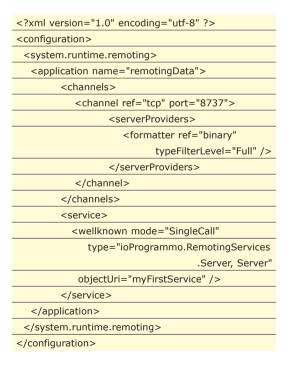
Configurando il tutto da codice potremmo ottenere lo stesso risultato, ma forse è più opportuno usare quest'altra forma decisamente più flessibile:

```
Server srv = (Server) Activator.GetObject(
                                        typeof(Server),
"tcp://localhost:8737/remotingData/myFirstService" );
```

A fare tutto il lavoro è l'oggetto System.Activator, che espone il metodo statico GetObject. Ad esso è sufficiente passare il tipo dell'oggetto da richiamare, ottenuto da una semplice typeof, in modo da consentirne il corretto marshaling, e l'indirizzo del servizio remoto che espone il tipo. Tale indirizzo è un normale URI e quindi è nella forma:

```
<channel>://<ip_o_indirizzo_host>:<porta>/
            <applicazione_remota>/<servizio_esposto>
```

Nel nostro caso il channel è il tcp che abbiamo appena registrato, l'indirizzo è localhost su porta 8737, come già definito nell'esempio introdotto nello scorso articolo. E sempre in quell'esempio avevamo definito l'applicazione remota "remotingData" e il servizio "myFirstService" che pubblicavano la classe Server. È evidente che tutto ciò che serve alla sua istanziazione è solo la stringa dell'URI e questo ci consente di rendere straordinariamente dinamica l'invocazione via remoting perché ogni volta che vogliamo cambiare l'indirizzo dell'oggetto non dobbiamo modificare il file .config, ma solo fornire una differente stringa all'Activator. Tale stringa, ad esempio, potrebbe essere ottenuta da un database e potrebbe variare in relazione ai diritti dell'utente o in base ad altri fattori quali il carico di elaborazione del o dei server o l'operare in locale o in remoto via Internet. Potremmo avere un servizio fisso che funga da Directory e cioè che, a richiesta, esponga le Url da utilizzare. Per cui un client invocherebbe sempre un metodo GetURI e questo risponderebbe con una URI dinamicamente determinata in base alla logica applicativa. Insomma un Object Broker estremamente semplice ma efficace. A questo punto siamo già in grado di invocare il servizio tenuto in piedi dall'host di esempio mostrato nello scorso articolo. Tale host era definito con il solito file di configurazione:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <system.runtime.remoting>
    <application name="remotingData">
        <channels>
            <channel ref="tcp" port="8737">
                <serverProviders>
                    <formatter ref="binary"
                                typeFilterLevel="Full" />
                </serverProviders>
            </channel>
        </channels>
        <service>
            <wellknown mode="SingleCall"
                type="ioProgrammo.RemotingServices
                                    .Server, Server"
                objectUri="myFirstService" />
        </service>
    </application>
  </system.runtime.remoting>
</configuration>
```

Ma se volessimo configurare anche l'host da codice, analogamente a quanto fatto per il client, dovremo aggiungere il riferimento all'assembly System.Runtime.Remoting.Dll e dichiarare con la using gli stessi namespace usati nel client. Inoltre dovremo anche riferire l'assembly contenente l'oggetto da esporre, operazione che nel caso dell'host con file di configu-

razione non si rende necessaria. A questo punto possiamo istanziare il BinaryFormatter lato server. Questa operazione è necessaria per impostare il parametro “typeFilterLevel” a “Full”, necessario per ottenere la bidirezionalità. Dunque istanziamo il channel nella version server e impostiamo la porta 8737 necessaria al funzionamento del nostro esempio:

```
IDictionary props = new Hashtable();
props["typeFilterLevel"] = "Full";
BinaryServerFormatterSinkProvider formatterProvider
 = new BinaryServerFormatterSinkProvider(props, null);
props = new Hashtable();
props["port"] = 8737;
props["name"] = "tcp";
TcpServerChannel channel = new
            TcpServerChannel(props, formatterProvider);
```

Dunque non ci resta che dichiarare e pubblicare come WellKnownServiceTypeEntity il nostro oggetto ioProgrammo.RemotingServices.Server. Il costruttore di questo oggetto richiede il tipo dell’oggetto da pubblicare, il nome del servizio col quale sarà pubblicato (il solito “myFirstService”) e la modalità di esecuzione:

```
WellKnownServiceTypeEntry entity =
                  new WellKnownServiceTypeEntry(typeof(
                  ioProgrammo.RemotingServices.Server),
                  "myFirstService", WellKnownObjectMode
                                              .SingleCall);
RemotingConfiguration.RegisterWellKnownServiceType
                                                  (entity);
```

Infine dichiareremo il nome dell’applicazione di remoting:

```
RemotingConfiguration.ApplicationName = "remotingData";
```

In pratica non abbiamo fatto altro che configurare tutte le parti della URI del servizio.

UTILIZZO DI INTERFACCE DI PROGRAMMAZIONE NELLE CHIAMATE REMOTE

La tecnica appena mostrata è davvero efficace in termini di flessibilità e di scalabilità del sistema, però impone ancora una rigidità evidente: il client e l’host devono condividere lo stesso assembly che contiene l’oggetto server. All’host serve perché evidentemente deve eseguirne il codice, ma al client serve semplicemente per conoscerne la struttura e cioè la firma dei metodi e delle proprietà. D’altro canto non è detto che il client sia in grado di far girare il componente server in locale perché magari questo accede ad un database o ad una qualsiasi altra risorse sw/hw che è fisicamente inaccessibile al client e quindi perché conservare l’intero assembly del server sul client solo per qualche firma? La soluzione è insita non nel modello di remoting ma nei principi dell’OOP stessa: il polimorfismo.
Questo si può ottenere in due forme: per ereditarietà o per interfaccia. Nel caso del remoting è decisamente più auspicabile il secondo modello visto che ciò che serve al client è solo la struttura della classe, non una sua implementazione di base. Inoltre l’assenza di ereditarietà multipla in NET e l’obbligo di derivare gli oggetti da remotizzare dalla classe MarshalByRefObject potrebbero introdurre delle limitazioni fastidiose se si optasse per il modello per ereditarietà. Dunque definiamo una semplice interfaccia che contenga l’insieme delle proprietà e dei metodi che il client invocherà sul server:

```
public interface IRemoteServer {
   string ScalarMethod(string par1, ref int par2);
   DataSet GetData();
}
```

Tale interfaccia sarà definita in un assembly terzo di scambio che sarà comune al client e al server. Esso, infatti, non conterrà alcuna logica implementativa ma solo l’insieme delle classi e delle interfacce necessaria alla corretta comunicazione tra client e server. A questo punto aggiungiamo nel progetto del server un riferimento all’assembly terzo e realizziamo una nuova classe server che implementa l’interfaccia IRemoteServer:

```
public class InterfaceServer : MarshalByRefObject,
                                       IRemoteServer  {
   #region IRemoteServer Members
   public string ScalarMethod(string par1, ref int par2) {
   //implementazione metodo ScalarMethod () }
   public DataSet GetData()  {
      //implementazione metodo GetData() }
   #endregion
}
```

Dal punto di vista del server non è dunque cambiato nulla e nemmeno dal punto di vista dell’host ci sono modifiche da apportare. Invece, è sul client che la differenza si fa sentire in modo evidente. Infatti il client non sarà più tenuto a conservare un riferimento all’assembly del server, che quindi non verrà più distribuito a tutti i client, ma dovrà soltanto conservare un riferimento all’assembly terzo che è assolutamente neutro, privo di implementazione, non direttamente eseguibile dal client nemmeno per errore e, soprattutto, privo di ogni riferimento ai dettagli implementativi del server o a risorse che esso usa. Ed infatti il client, per invocare il server, si

limiterà a scrivere:

```
IRemoteServer srv = (IRemoteServer)
        Activator.GetObject(typeof(IRemoteServer),
    "tcp://localhost:8737/remotingData
                        /myFirstService" );
```

Oltre all'indubbio vantaggio di non dover distribuire alcun componente server insieme ai client, questa tecnica polimorfica ci consente, in abbinamento ad un servizio di directory come quello esposto nei primi paragrafi, di far puntare il client a server completamente diversi, su macchine diverse e, soprattutto, con classi implementative diverse che hanno semplicemente in comune il fatto di implementare l'interfaccia IRemoteServer. Gli scenari che si aprono sono assolutamente vasti così come le possibilità di tuning e di configurazione delle proprie applicazioni distribuite.

COMUNICAZIONE BIDIREZIONALE

Il modello tradizionale di programmazione distribuita, implica che sia sempre un client a contattare in modalità sincrona il server e mai viceversa. Questo approccio è sostanzialmente sempre corretto, però in qualche caso può non risultare sufficiente. Si supponga, ad esempio, di voler sottoscrivere degli eventi del server (disconessioni, riavvii per manutenzione, ecc…) e cioè tutte condizioni in cui deve essere il server a segnalare o a richiedere qualcosa ai client e non viceversa. In realtà anche queste situazioni possono essere gestite col tradizionale modello unidirezionale, magari con i client che effettuano un polling cadenzato al server per verificare che questi non abbia da comunicargli qualcosa. Certo non è elegante, ma se implementato bene può funzionare. Fortunatamente, però, l'architettura di remoting offre la possibilità di una comunicazione vera tra server e client. Per far questo è necessario che il client sia disposto ad accettare richieste provenienti dal server. Come questo viene implementato dipende fortemente dal channel utilizzato. Ad esempio il channel http non consente questo modello di bidirezionalità. Invece, il channel Tcp fornito da Microsoft lo consente, a patto però di aprire una nuova porta TCP sul client per ricevere le richieste dal server. Dunque due canali aperti su due porte differenti: sulla prima il client contatta il server e sulla seconda il server contatta il client. Non è molto efficace ed è piuttosto pretenzioso in termini di richieste all'amministratore di rete, però funziona e in qualche caso può risolvere un sacco di problemi di programmazione. Vediamo, usando il channel TCP, quali sono gli accorgimenti da adoperare per realizzare questa forma di bidirezionalità. Sull'host, come già accennato in precedenza, dobbiamo impostare il BinaryFormatter col parametro TypeFilterLevel a "Full":
con file di configurazione:

```
<serverProviders>
    <formatter ref="binary" typeFilterLevel="Full" />
      </serverProviders>
```

oppure, se da codice:

```
IDictionary props = new Hashtable();
props["typeFilterLevel"] = "Full";
BinaryServerFormatterSinkProvider formatterProvider =
    new BinaryServerFormatterSinkProvider(props, null);
```

Invece sul client dobbiamo limitarci ad indicare la porta TCP sulla quale ricevere le comunicazioni dal server. Se non intendiamo fissarne una particolare, ma far scegliere al client al momento della richiesta, impostiamo semplicemente la porta a 0:
con file di configurazione:

```
<channel ref="tcp" port="0">
```

oppure, se da codice:

```
IDictionary props = new Hashtable();
props["port"] = 0;
TcpChannel channel = new TcpChannel(props,
                        formatterProvider, null);
```

Siamo così a pronti a ricevere comunicazioni dal server. Tali comunicazioni possono essere degli eventi del server che il client ha sottoscritto o anche semplici richieste che il server fa al client per ragioni applicative. Perché il server possa contattare il client via remoting, anche sul client devono essere presenti oggetti con caratteristiche di remotizzabilità. Ed infatti nel nostro assembly comune tra client e server definiamo un oggetto client di remotizzazione che verrà usato dal server per effettuare le sue segnalazioni al client:

```
//Oggetto client definiti nell'assembly comune
public class ClientObject : MarshalByRefObject {
  public delegate void MessageEvent (object sender,
                        RemotingEventArgs e);
  public event MessageEvent SyncRemotingEvent;
  public event MessageEvent AsyncRemotingEvent;
  public virtual void OnAsyncServerNotification(
                        RemotingEventArgs e) {
     try {
        if (AsyncRemotingEvent != null)
           AsyncRemotingEvent.BeginInvoke(
                        this, e, null, null); }
     catch { } }
  public virtual void OnSyncServerNotification(
```

```
                                  RemotingEventArgs e) {
        try {
            if (SyncRemotingEvent != null)
                          SyncRemotingEvent(this, e); }
        catch { }
    } }
//oggetto eventargs che conterrà le informazioni sull'evento
//scatenato anch'esso è definito nell'assembly comune
[Serializable]
public class RemotingEventArgs : EventArgs {
    private int _code;
    private string _message;
    public RemotingEventArgs(int code, string
                                    message) : base() {
        _code = code;
        _message = message;}
    public int EventCode {
        get {return _code; }
    }
    public string Message  {
        get {return _message; }
    } }
//nuova interfaccia IRemoteServer contentente i
//metodi per passare il riferimento
//all'oggetto ClientObject al server
public interface IRemoteServer
{ string ScalarMethod(string par1, ref int par2);
    DataSet GetData();
    void TestAsyncBidirectionComm ( ClientObject
                            client, string message );
    void TestSyncBidirectionComm ( ClientObject client,
                                       string message );
}
```

Ed ecco le implementazioni dei due metodi Test nella classe InterfaceServer:

```
public void TestAsyncBidirectionComm(ClientObject
                            client, string message) {
    client.OnAsyncServerNotification(new
                RemotingEventArgs(343, message));
}
public void TestSyncBidirectionComm(ClientObject
                            client, string message) {
    client.OnSyncServerNotification(new
                RemotingEventArgs(341, message));
}
```

Molto semplicemente, quando il client invoca il server passa anche un riferimento all'istanza di ClientObject; il server immediatamente invocherà il metodo OnSyncServerNotification o OnAsyncServerNotification del client aprendo, quindi, la comunicazione nella direzione opposta. Il codice del client sarà altrettanto semplice:

```
private void cmdTestBiSynk_Click(object sender,
                            System.EventArgs e) {
    ClientObject client = new ClientObject();
    ClientObject.MessageEvent handler = new
                          ClientObject.MessageEvent(
                          ServerRequestEventHandler);
    client.SyncRemotingEvent += handler;
    IRemoteServer srv = (IRemoteServer) Activator
                   .GetObject(typeof(IRemoteServer),
     "tcp://localhost:8737/remotingData/myFirstService" );
    srv.TestSyncBidirectionComm(client, "Sono il server!");
    MessageBox.Show("Sono il client!"); }
public void ServerRequestEventHandler(object sender,
                                  RemotingEventArgs e) {
    MessageBox.Show(e.Message);
}
```

Il client crea un'istanza di ClientObject, l'evento SyncRemotingEvent con l'event handler ServerRequestEventHandler, quindi istanzia il server e ne invoca il metodo TestSyncBidirectionComm passando il riferimento all'oggetto ClientObeject appena creato. Dopo qualche istante il server invocherà l'evento MessageEvent ripassando, quindi, la palla al client. La **Figura 1** mostra con maggiore chiarezza questo meccanismo.

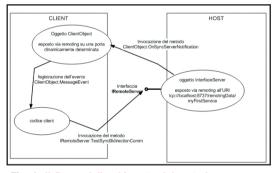



Fig. 1:* *Il flusso della chiamata del metodo TestSyncBidirectionalComm

Esiste, però, un potenziale problema: nell'event handler c'è una messagebox che quindi è bloccante perché viene atteso il click dell'utente. Questo significa che il server resta in attesa della messagebox del client e, nell'ipotesi in cui il server sia una Singleton, ciò significherebbe che tutti i client resterebbero in attesa del click della messagebox su uno dei client! Questo è inacettabile, ma per fortuna in .NET esiste un ottimo supporto alle chiamate asincrone: in pratica possiamo fare in modo che il server invochi l'evento del client, ma non resti in attesa della risposta e che quindi lo invochi in forma asincrona. Per far ciò ci vengono in aiuto i delegate; ed ecco infatti che l'istuzione:

```
if (SyncRemotingEvent != null)
                        SyncRemotingEvent(this, e);
```

diventa, con la BeginInvoke del delegate che si nasconde dietro l'evento:

L'AUTORE

Vito Vessia è amministratore e cofondatore della codeBehind S.r.l. (www.codeBehind.it), una software factory di applicazioni enterprise, web e mobile, dove progetta e sviluppa applicazioni e framework in .NET, COM(+) e Delphi occupandosi degli aspetti architetturali. È autore del libro "Programmare il cellulare", Hoepli, 2002, sulla programmazione dei telefoni cellulari connessi al PC con protocollo standard AT+. Può essere contattato tramite e-mail all'indirizzo vvessia@katamail.com

```
if (AsyncRemotingEvent != null)
  AsyncRemotingEvent.BeginInvoke(this, e, null, null);
```

L'effetto è facilmente osservabile eseguendo il programma.

CONCLUSIONI

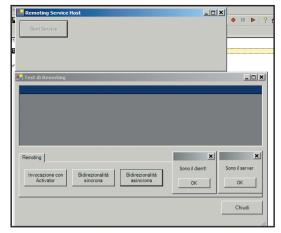
Si conclude così questa lunga disamina della tecnologia .NET Remoting. Nella **Figura 2** possiamo apprezzare la nostra applicazione di prova in esecuzione. Abbiamo avuto modo di conoscere in modo piuttosto approfondito la tecnologia e così adesso siete pronti ad utilizzarla in contesti reali e non solo in esempi gradevoli, ma poco utili nella realtà. Di questa tecnologia si potrebbe discutere ancora lungamente. Ad esempio si potrebbe mostrare come utilizzare channel di terze parti che migliorano le performance e le caratteristiche dei componenti originali Microsoft o, addirittura, come scrivere un proprio channel o altre parti dell'architettura di remotizzazione. Ma forse saranno argomento di prossimi articoli, intanto non vi resta che cominciare a pensare le vostre applicazioni in modo distributo: adesso avete un formidabile strumento in più per realizzarle.

Vito Vessia

Fig. 2: *Il risultato: il nostro client in esecuzione*

MODALITÀ SINGLE CALL E SINGLETON

Un oggetto server di remoting può essere attivato in due modi:

- **SINGLE CALL (ogni richiesta dei client provoca una nuova istanziazione di un oggetto sul server che risponde all'invocazione del client e muore immediatamente dopo e questo accade per richieste consecutive dello stesso client);**
- **SINGLETON (tutte le richieste di tutti i client vengono soddisfatte dalla stessa istanza dell'oggetto server remoto).**

La differenza è sostanziale e deve essere tenuta fortemente in conto quando si decide di implementare un'architettura distribuita nelle proprie applicazioni usando .NET Remoting. Si osservi infatti questo esempio. Si supponga di voler istanziare l'oggetto server:

```
Server remoteObject = new Server();
```

Dopo questa istruzione l'oggetto server non è ancora stato contattato per cui nessuna istanza sul server viene creato né tanto meno viene controllato che il server sia in grado di rispondere. Soltanto all'invocazione del metodo la richiesta giunge effettivamente al server:

```
DataSet ds = remoteObject.GetData();
```

Inoltre si supponga che l'oggetto Server disponga di una proprietà Name di tipo r/w e si osservi il codice in tabella. Il comportamento di tale codice, con l'oggetto Server che gira in remoting, è completamente diverso rispetto al suo omologo in locale e anche nella versione Single Call e Singleton ci sono differenze consistenti. I vantaggi del Singleton sono evidenti in termini di semplificazione della programmazione. Esso infatti consente di mantenere gli stati interni da una chiamata all'altra (stateful) e inoltre fa da memoria centrale per tutte le richieste. Con un singleton, infatti, potrete conservare tutte le informazioni di tutti i clienti e avere sempre un'idea precisa di quanto sta accadendo nel vostro sistema software visto che ogni richiesta passa dal singleton e questo è in grado di tenerne traccia e di ricordarsene. Lo svantaggio è evidente: se il singleton va giù tutti i client vanno giù... Il Single Call, d'altro canto, impone una programmazione senza stati (stateless) in cui tutto ciò che serve a rispondere ad una richiesta di un client deve essere passato come parametri del client stesso o deve essere reperito dal server in altre forme (ad esempio da un database) e non si può programmare sperando che sia sempre la stessa istanza a rispondere a conservare le impostazione della chiamata precedente. Questo modello in compenso fornisce una scalabilità ed una robustezza ben maggiori. Dunque a voi la scelta. Probabilmente un modello misto si dimostra vincente in quasi tutte le circostanze.

Istruzione	Versione Locale	Versione remota Single Call	Versione remota Singleton
Server firstObject = new Server();			
firstObject.Name = "Alice";			
MessageBox.Show(firstObject.Name);	restituisce *"Alice"*	restituisce *null*	Restituisce *"Alice"*
Server secondObject = new Server();			
MessageBox.Show(secondObject.Name);	restituisce *null*	restituisce *null*	restituisce *"Alice"*
secondObject.Name = "Beatrice";			
MessageBox.Show(secondObject.Name);	restituisce *"Beatrice"*	restituisce *null*	restituisce *"Beatrice"*
MessageBox.Show(firstObject.Name);	restituisce *"Alice"*	restituisce *null*	restituisce *"Beatrice"*

Excel per motore C# per pilota

In questo articolo impareremo ad usare un foglio di lavoro Excel attraverso un software scritto in C#. Vedremo come estrarne dati e trasformarli inserendoli in un Database

Excel è uno degli strumenti di Microsoft Office più utilizzato per l'analisi dei dati. Sebbene il suo punto di forza sia quello di poter effettuare calcoli anche abbastanza complessi, nella maggior parte dei casi è utilizzato per creare delle semplici tabelle utili a raggruppare e visualizzare una serie di dati, senza sfruttarne tutta la potenza.
In questo articolo vedremo come usare un foglio Excel da un software scritto in .Net (C#), capiremo quanto questa accoppiata sia utile e quante grane può risolvere.

PREMESSA

Nella maggior parte delle applicazioni che vengono sviluppate, si ha la necessità di archiviare delle informazioni e dei dati per un successivo riutilizzo. Si pensi ad un qualsiasi sito web che richiede la registrazione degli utenti. Le informazioni quali nome utente, password, indirizzo di posta elettronica vengono normalmente registrate in un DataBase per poi essere utilizzate, ad esempio, per il logon al sito o per l'invio di una newsletter.
Sebbene un DataBase ci consenta di archiviare ed usare i dati registrati in maniera abbastanza semplice e veloce, spesso non è lo strumento migliore per effettuare un'analisi manuale delle informazioni in esso contenute. Nel caso si stia utilizzando Microsoft Access sarebbe eventualmente possibile realizzare dei report o delle maschere *(forms)* che presentino le informazioni in maniera leggibile ma, in presenza di DataBase più evoluti come Sql Server, Oracle ecc., l'operazione non è altrettanto semplice.
Si ricorre quindi a strumenti diversi e, uno degli strumenti maggiormente usati è Microsoft Excel.

UNA PANORAMICA DI EXCEL

Excel è un potente foglio di calcolo prodotto da Microsoft. Generalmente è rilasciato insieme alla suite di programmi per ufficio Microsoft Office, attualmente giunta alla versione 2003. Un foglio di lavoro di Excel è organizzato per righe e colonne la cui intersezione definisce una cella.
Alle celle di un foglio di lavoro ci si riferisce per nome della colonna e numero della riga. La prima cella in alto a sinistra del foglio di lavoro sarà quindi la cella *A1*. Teoricamente non c'è un limite al numero di righe e colonne che ogni foglio può contenere.
L'insieme dei fogli di lavoro di Excel costituisce la cartella di lavoro. Come per righe e colonne, non c'è teoricamente limite al numero di fogli che una cartella può contenere.
Ogni cella può contenere svariati tipi di dati: testo, numeri, valute, data ed ora e, se questo non dovesse bastare, è possibile definire dei formati personalizzati.

LA LOGICA DI COLONNE E RIGHE

Uno dei punti di forza del foglio di lavoro di Excel è quello di poter eseguire calcoli all'interno delle celle. Inserendo infatti il simbolo = all'interno di una cella, sarà possibile eseguire calcoli anche abbastanza complessi i cui valori possono essere recuperati da altre celle del foglio. Ad esempio, se volessimo effettuare la somma dei valori contenuti nelle celle *A1* e *B1*, nella cella *C1* scriveremo =*Somma (A1:B1)*.
Per avere una idea delle operazioni che è possibile eseguire all'interno di una cella, è possibile aprire l'editor delle funzioni dal menu inserisci e selezionando la voce funzione.

La visualizzazione "tabellare" dei dati, la possibilità di inserire formule e grafici, il numero illimitato di righe e colonne, fanno di Excel uno strumento molto comodo e potente per l'analisi dei nostri dati.

"GIOCHIAMO" CON EXCEL

Nel precedente paragrafo abbiamo visto, in maniera molto sintetica, quali sono le potenzialità di Microsoft Excel. In questo paragrafo inizieremo a manipolare un foglio di lavoro da un programma scritto in C#.
Vedremo come aprire un file, modificare il contenuto di una cella, inserire dei valori, modificarne il formato ed infine salvare il lavoro.
Questo esempio ci servirà ad iniziare a prendere confidenza con il modello di programmazione di Microsoft Excel; tutto quello che apprenderemo in questo paragrafo ci servirà successivamente per esaminare un caso reale di applicazione.
Apriamo il nostro Visual Studio, creiamo un nuovo progetto di tipo *"Applicazione per Windows"* ed aggiungiamo immediatamente un riferimento a Microsoft Excel. Per farlo, clicchiamo con il tasto destro del mouse sulla cartella References e selezioniamo la voce *"Aggiungi riferimento"*.
Dalla maschera che si aprirà, attiviamo la scheda *COM* e selezioniamo la voce *"Microsoft Excel xx object Library"*, dove *xx* è la versione di Excel che avete installato sui vostri PC.
Una volta referenziati gli elementi di Microsoft Excel, possiamo iniziare a lavorarci su.
Il primo passo è quello di istanziare un nuovo oggetto di tipo *Excel.Application* che chiameremo *xlsApp*. Attraverso esso avremo la possibilità di accedere agli elementi del foglio excel, prima tra tutti la cartella di lavoro *(Excel.Workbook)* che nella nostra applicazione si chiama *xlsWb*.
Abbiamo l'applicazione la cartella di lavoro, manca solo il foglio su cui lavorare: *Excel.Worksheet (xlsWs)*. Vediamo un esempio pratico:

```
Excel.Application xlsApp = new Excel.Application();
Excel.Workbook xlsWb = xlsApp.Workbooks.Open(
    Environment.CurrentDirectory+@"\test.xls", 0, false,
    5, "", "", true, Excel.XlPlatform.xlWindows, "\t",
    false, false, 0, true, true, false);
Excel.Worksheet xlsWs =
    (Excel.Worksheet)xlsWb.ActiveSheet;
```

Per prima cosa istanziamo un oggetto di tipo .

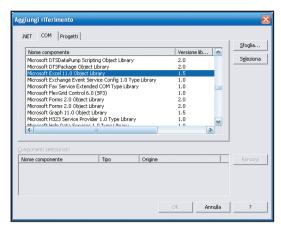



Fig. 1: ***Aggiunta del riferimento alle librerie di Excel nel nostro software***

L'istanza di questo oggetto ci darà successivamente accesso a tutti gli elementi del foglio Excel su cui andremo a lavorare.
Creiamo dunque una istanza di *Excel.Workbook (xlsWb)* per accedere alla cartella di lavoro e *Excel.Worksheet (xlsWs)* per accedere al foglio vero e proprio.

```
xlsWs.get_Range("A1", "C1").Font.Bold = true;
    //Rendiamo grassetto il testo delle prime 3 celle
xlsWs.get_Range("A1", "C1").Borders.LineStyle =
    Excel.XlLineStyle.xlContinuous; //Aggiungiamo
    un bordo alle 3 celle
xlsWs.get_Range("A1", "A1").Value2 = "Nome";
xlsWs.get_Range("B1", "B1").Value2 = "Cognome";
```

Tutte le caratteristiche del prodotto Microsoft Excel sono accessibili a partire da questo sito: http://www.microsoft.com/italy/office/excel/prodinfo/overview.mspx.
Per tutti gli altri prodotti della famiglia Micrsofto Office invece, è possibile partire da qui: http://www.microsoft.com/italy/office

DA E VERSO EXCEL. PERCHÉ?

Mi è capitato spesso di dover spiegare il perché, in determinate situazioni, è necessario muoversi da Microsoft Excel ad un Data Base e perché spesso vale anche il discorso contrario. La prima obiezione che viene fatta è: "uso il foglio di lavoro per organizzare i miei dati. Che bisogno ho di un Data Base?" e, nell'altro verso, "ma se ho già un Data Base, perché devo usare Excel?".
Dare una risposta non è sempre semplice in quanto la scelta di un sistema piuttosto che un altro viene molto influenzata dalla quantità e dal tipo di dati da trattare. I DataBase, come tutti sappiamo, servono ad immagazzinare informazioni in maniera strutturata.
Sebbene questa metodologia introduca notevoli vantaggi (eliminazione della ridondanza dei dati, organizzazione logica, univocità ecc.), il dato archiviato perde spesso di concretezza. Non è raro infatti imbattersi in tabelle composte da soli indici!
Tali dati possono essere analizzati solo dopo essere stati nuovamente aggregati attraverso opportune query, viste o stored procedure ed uno dei metodi più comuni per presentarli è appunto quello la modalità tabellare. Disporre di uno strumento che ci consente di muoverci da una piattaforma ad un'altra senza complesse procedure di porting, diventa estremamente vantaggioso per noi sviluppatori, in quanto potremmo creare software più vicini alle esigenze dei nostri clienti.

Su MSDN è disponibile un interessante articolo dal titolo:

- **UNDERSTANDING THE EXCEL OBJECT MODEL FROM A .NET DEVELOPER'S PERSPECTIVE**

in cui viene spiegato il modello ad oggetti di Excel visto da .net. L'articolo comprensivo dei sorgenti è disponibile a questo indirizzo: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/odc_vsto2003_ta/html/excelobj.asp

```
xlsWs.get_Range("C1", "C1").Value2 = "Età";
xlsWs.get_Range("A2", "A2").Value2 = "Michele";
xlsWs.get_Range("B2", "B2").Value2 = "Locuratolo";
xlsWs.get_Range("C2", "C2").Value2 = "28";
```

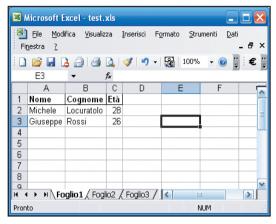



Fig. 2: *Il risultato del nostro primo software*

Usiamo *xlsWs.get_Range()* per selezionare un insieme di celle (nell'esempio la selezione va dalla cella *A1* alla cella *C1*). Sul range di celle attivato possiamo compiere tutte le operazioni che faremmo manualmente direttamente sul foglio come definire dei formati, inserire del testo ecc.

```
xlsWb.Save();
xlsWb.Close(false, "", false);
```

Quando abbiamo finito, salviamo il nostro file Excel e, cosa molto importante, ricordiamoci di chiuderlo.
La nostra prima applicazione è completa. Sebbene sia estremamente semplice, lo scopo è quello di chiarire alcuni passi fondamentali ed indispensabili prima di lavorare su applicazioni reali.

QUALCHE TRASFORMAZIONE

Nel precedente paragrafo abbiamo visto come è possibile interagire con dei fogli Excel attraverso un software scritto usando il .Net Framework.
Sappiamo che tutti i produttori di RDBMS

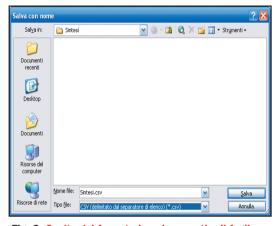
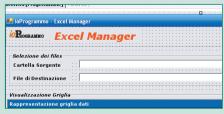



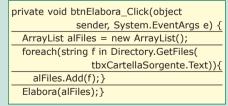
Fig. 3: *Scelta del formato in cui convertire il foglio Excel*

EXCEL E C#: REAL WORLD

Supponiamo di avere degli agenti che stanno compiendo un sondaggio di gradimento sulla nostra azienda. Forniremo ai nostri collaboratori un modello excel che dovranno compilare. Alla fine del lavoro ci reinvieranno tutte le schede in formato elettronico che dovremo elaborare singolarmente. È evidente che, sebbene un modello così fatto sia di facile lettura, non è di certo il formato ideale per una elaborazione elettronica. Il nostro scopo sarà quello di trasformare i dati dalla rappresentazione del modello in un unica tabella, più semplice da gestire attraverso dei processi automatizzati

1 **In Visual Studio, dopo aver aggiunto i riferimenti a Microsoft realizziamo un form come in figura. Esso conterrà i campi per selezionare la cartella in cui risiedono i files, un campo in cui specificare il foglio di sintesi, una griglia (DataGrid) per la rappresentazione dei dati e un log immediato.**

```
folderBrowserDialog1.SelectedPath =
        Environment.CurrentDirectory;
folderBrowserDialog1.ShowDialog();
tbxCartellaSorgente.Text =
        folderBrowserDialog1.SelectedPath;
ShowFileList();
```

2 **Aggiungiamo al nostro form un elemento FolderBrowserDialog. Questo ci consentirà di esplorare le cartelle del nostro sistema e di selezionarne una. Una volta eseguita la selezione, il tipo di ritorno della FolderBrowserDialog sarà una stringa che rappresenta il path completo della cartella che nel nostro caso conterrà i file con i sondaggi compilati.**

```
private void btnElabora_Click(object
        sender, System.EventArgs e) {
  ArrayList alFiles = new ArrayList();
  foreach(string f in Directory.GetFiles(
        tbxCartellaSorgente.Text)){
    alFiles.Add(f);}
  Elabora(alFiles);}
```

3 **L'elaborazione dei nostri files si divide in due step. Il primo consiste nella creazione di un elenco dei files da elaborare (alFiles). Questo vettore conterrà il path completo di ogni singolo file da passare al metodo che si occuperà di elaborarlo. Il secondo step è appunto quello di passare l'array al metodo opportuno (Elabora(alFiles)).**

danno la possibilità di importare dati da un foglio di testo, i cui campi sono separati da un delimitatore standard. Excel ha la possibilità di trasformare un foglio di lavoro direttamente in un file di testo delimitato. Tale funzione è accessibile dal menu *File*, scegliendo la voce *Salva con Nome* e selezionando l'opportuno formato. Come abbiamo detto nei precedenti paragrafi, l'oggetto *Excel.Application* che istanziamo nel nostro programma ci consente di fare, via codice, esattamente quello che facciamo manualmente direttamente nel file.
Per esportare il nostro foglio di sintesi in formato testo delimitato ci basta fare:

```
private void btnExportToTxt_Click(object sender,
                              System.EventArgs e){
  Excel.Workbook xlsWbSintesi = null;
  xlsAppSintesi = new Excel.ApplicationClass();
  xlsWbSintesi = xlsAppSintesi.Workbooks.Open(
     tbxFileDestinazione.Text, 0, false, 5, "", "", true,
        Excel.XlPlatform.xlWindows, "\t", false, false,
                                  0, true, true, false);
xlsWbSintesi.SaveAs(tbxCartellaSorgente.Text+@"
        \sintesi\sintesi.txt", Excel.XlFileFormat.xlCSV,
  Type.Missing, Type.Missing, Type.Missing, Type.Missing,
   Excel.XlSaveAsAccessMode.xlNoChange, Type.Missing,
 Type.Missing, Type.Missing, Type.Missing, Type.Missing);
  xlsWbSintesi.Close(false, "", false);
}
```

ed il nostro file sarà pronto per essere importato in un Data Base.

Nel codice allegato è presente anche una funzione che trasforma il file di testo in un DataSet visualizzabile poi nella DataGrid. Avendo a disposizione un DataSet diventa estremamente semplice passarlo ad esempio ad un servizio web remoto che si occuperà di gestirlo.

CONCLUSIONI

Uno dei punti di forza dei sistemi Microsoft è l'alta integrazione di tutti i software appartenenti a questa piattaforma. Fino a qualche tempo fa il costo di quest'alta integrazione era da pagarsi in termini di oggetti com, e dll, cosa che comportava una qualche instabilità e comunque un overhead del sistema.
L'avvento di .NET aggira questo ostacolo, rendendo l'integrazione molto più efficace. In questo articolo abbiamo visto come con semplici passi, usare dei fogli di lavoro Microsoft Excel attraverso dei software scritti usando il *.net Framework*.
Excel è uno strumento di analisi molto potente e, per i meno esperti, è decisamente più semplice e comodo dell'utilizzo di un Data Base. Poter "muovere" dati da e verso il foglio di lavoro in modo del tutto automatizzato ci consente di risparmiare gran parte del tempo che si sprecherebbe per rifare le stesse operazioni manualmente.

Michele Locuratolo

Per una guida sulle formule utilizzabili in Excel, il punto di riferimento è il sito http://office.microsoft.com/it-it/assistance/CH062528031040.aspx **mentre per le funzioni si può partire da qui:** http://office.microsoft.com/it-it/assistance/CH062528191040.aspx.

L'autore può essere contattato attraverso il suo blog su http://blogs.mindbox.it/mighell **e sarà lieto di rispondere alle domande dei lettori.**

```
Excel.Workbook xlsWbSintesi = null;
xlsAppSintesi = newExcel.ApplicationClass();
xlsWbSintesi = xlsAppSintesi.Workbooks
                .Open(tbxFileDestinazione.Text,
                  0, false, 5, "", "", true,
                     Excel.XlPlatform.xlWindows,
                                  "\t", false,
                           false,  0, true, true,
                                        false);
Excel.Worksheet xlsWsSintesi = (
                              Excel.Worksheet)
                        xlsWbSintesi.ActiveSheet;
```

4 **Tra le prime cose da fare nell'elaborazione dei files è quella di aprire il file che raccoglierà I dati di sintesi. In questo modo infatti, verrà aperto una sola volta (prima dell'inizio del ciclo di elaborazione dei singoli files) e chiuso alla fine dell'elaborazione. Questo ci consente di risparmiare tempo che altrimenti impiegheremo in operazioni di I/O.**

```
foreach(string Filename in files){
  Excel.Workbook xlsWbSondaggio = null;
  xlsAppSondaggio = new Excel.Application();
  xlsWbSondaggio = xlsAppSondaggio
   .Workbooks.Open(Filename, 0,true, 5, "",
   "", true, Excel.XlPlatform.xlWindows, "\t",
          false, false, 0, true, true, false);
  Excel.Worksheet xlsWsSondaggio = (Excel
     .Worksheet)xlsWbSondaggio.ActiveSheet;
  Nome = xlsWsSondaggio.get_Range("B2",
                     "B2").Text.ToString();
  Cognome   = xlsWsSondaggio.get_Range(
                "D2", "D2").Text.ToString();
```

5 **Il path di ogni file Excel da elaborare è contenuto nell'array che abbiamo passato al metodo di elaborazione. Dobbiamo quindi aprire ogni foglio Excel, recuperare i dati che ci servono (che per comodità sono stati associati a delle variabili) per poi successivamente inserirli nel file di sintesi che abbiamo aperto nello step precedente.**

```
xlsWsSintesi.get_Range("A"+lastRow,
            "A"+lastRow).Value2 = Nome;
xlsWsSintesi.get_Range("B"+lastRow,
         "B"+lastRow).Value2 = Cognome;
xlsWsSintesi.get_Range("C"+lastRow,
        "C"+lastRow).Value2 = Indirizzo;
xlsWsSintesi.get_Range("D"+lastRow,
             "D"+lastRow).Value2 = CAP;
xlsWsSintesi.get_Range("E"+lastRow,
           "E"+lastRow).Value2 = Città;
lastRow ++;
} //Chiusura del ciclo foreach
xlsWbSintesi.Close(true,
         tbxFileDestinazione.Text, false);
```

6 **Sempre all'interno del ciclo foreach, scriviamo il valore delle nostre variabili di appoggio (che contengono i dati dei files sorgenti) nel foglio Excel di sintesi. Alla fine di ogni ciclo incrementiamo la variabile lastRow in modo da passare alla riga successiva. Completato il ciclo, salviamo e chiudiamo il file di sintesi.**

Analisi ai vertici dei VideoGames

Impariamo come sfruttare al massimo l'hardware delle moderne schede grafiche utilizzando i motori di Vertex Shader. Una tecnica che ci consentirà di realizzare VideoGiochi molto veloci ed elaborati

Molti di voi sapranno che gli oggetti che compongono una scena tridimensionale in un videogame vengono rappresentati utilizzando la composizione di molti triangoli. Ad esempio una sfera tridimensionale vista al microscopio apparirebbe composta da milioni di triangoli uniti tra loro che danno l'idea della sfera. Ora, volendo deformare la sfera per disegnare ad esempio un naso, sarebbe necessario deformare ogni singolo triangolo che la compone al fine di ottenere la forma desiderata. È anche vero che i triangoli sono composti da vertici, e che ciascun vertice possiede proprie caratteristiche che lo definiscono, oltre alla posizione *x,y,z* è dotato ad esempio di attribuito che ne definiscono il colore, la luminosità, eventuali texture. È facile rendersi conto che qualunque effetto si voglia applicare a una forma tridimensionale passa per la modifica degli attributi dei vertici dei triangoli che compongono la forma. Questo implica per ogni singolo "frame" di un'animazione milioni di calcoli che in tempi passati erano talmente complessi da essere quasi impossibili. Immaginate che l'effetto più semplice, ovvero applicare una luce su una superficie corrisponde a ridisegnare tutti i vertici dei triangoli che compongono la superficie. Se poi si vuole ottenere ad esempio un effetto nebbia in movimento, il numero di calcoli diventa impressionante, se poi vogliamo per caso deformare contemporaneamente o far muovere un oggetto tutto diventa molto complesso. Effetti del genere non erano possibili in passato. La situazione è radicalmente cambiata con l'avvento dei processori HardWare inseriti direttamente nelle schede grafiche e che fanno da supporto alla CPU principale del computer. In questo modo si riesce a delegare una parte dei calcoli ai processori delle schede grafiche, che espongono funzioni ottimizzate proprio per realizzare effetti grafici complessi. Questo genere di processori prende il nome di *"Vertex Shader"*.

COME INIZIARE

È necessario avere installato il DirextX SDK. Nessun altro componente è richiesto. Se volete provare da soli a realizzare dei modelli 3D, uno strumento piuttosto interessante è Blender. Si tratta di un modeller 3D OpenSource. Riprodurre gli esempi proposti in questo articolo è molto semplice. È sufficiente utilizzare il programma "EffectEdit" contenuto nell'SDK per le DirectX. Si tratta di un simulatore per la gestione dei VS. È sufficiente digitare il codice, salvarlo in un normale file di testo e caricarlo dentro EffectEdit per simulare i vostri effetti.

I VERTEX SHADER E LE DIRECTX

Ciascun costruttore di Hardware ha integrato all'interno delle proprie schede un processore *"Vertex Shader"*. Questo non vuol dire che per ogni scheda grafica debba essere sviluppato un codice particolare per accedere alle funzionalità esposte dal processore. A partire dalla versione 8.0 delle DirectX è stato elaborato uno standard per il Vertex Shader, perciò utilizzando le DirectX si ha la certezza di scrivere codice perfettamente compatibile per tutti i processori VS delle varie schede video.

LE MANI NEL GIOCO

Fino a qualche tempo fa le funzioni esposte dai processori VS erano statiche. Bisognava cioè accontentarsi di quanto messo a disposizione dalla scheda. Ovvero, se la scheda esponeva l'effetto nebbia, il programmatore poteva utilizzarlo nelle proprie applicazioni, altrimenti no. Inoltre l'effetto era programmato dai realizzatori del VS e

non dal programmatore del videogioco. Da qualche anno a questa parte gli shader supportati dalla schede video sono di tipo "programmabile". Questo vuol dire che è possibile applicare virtualmente qualsiasi tipo di effetto mediante shader, senza limitarsi, come si faceva prima, a quelli predefiniti. Inoltre il fatto che gli shader siano manipolati direttamente dall'hardware della scheda video, li rende molto veloci, senza impatti dannosi sulla velocità di rendering della scena. È possibile programmare i VS utilizzando un linguaggio che acceda direttamente all'hardware, una sorta di assembler dedicato al 3D.
Ovviamente un linguaggio di questo tipo è poco "maneggevole". Per questo sono stati pensati diversi linguaggi di "alto livello" che permettano di scrivere codice in maniera più leggibile e più vicina a un linguaggio di programmazione tradizionale. Parliamo di *Cg (= "C for Graphics")* di NVidia e *HLSL (= "High Level Shading Language")* di Microsoft. Le differenza tra questi linguaggi sono davvero minime e dettate da motivi commerciali più che tecnici.

UTILIZZIAMO DIRECT3D E HLSL

È possibile programmare un effetto VS inserendo il codice in un semplice file testuale che abbia estensione *".fx"*. Il codice siffatto può essere utilizzato così com'è nella nostra applicazione, chiamando le opportune funzioni di Direct3D.
Per potere provare l'effetto senza dovere lanciare però tutta l'applicazione che stiamo sviluppando possiamo utilizzare *EffectEdit*. *EffectEdit* è un piccolo tool presente nell'ambiente di sviluppo DirectX (il DirectX 9.0 SDK) che consente di testare e modificare "al volo" il codice dell'effetto.
Vengono fornite da questo tool diverse agevolazioni, che consentono di velocizzare di molto il lavoro. Ad esempio le seguenti righe di codice:

```
// modello 3D
string XFile = "airplane\\airplane 2.x";
// direzione della luce
float3 DirFromLight < string UIDirectional = "Light
                                          Direction"; >
= {0.577, -0.577, 0.577};
```

consentono di caricare un modello 3D nel formato *".x"* di Direct3D. Il trucco utilizzato da *EffectEdit* è semplice: si dichiara una stringa con il nome particolare *"XFile"* e questa viene utilizzata come modello al posto della mesh di default (una sfera). Questa stringa non avrà alcun effetto sul VS quando questo verrà utilizzato all'interno della nostra applicazione. Stesso discorso per *DirFromLight* che è un vettore composto da 3 float: inizializzandolo con la stringa *UIDirectional*, EffectEdit creerà a schermo un vettore mobile che consente di cambiare in maniera facile la direzione della luce.

INIZIALIZZAZIONI

Il codice vero e proprio dello shader che andremo ad analizzare comincia con la definizione delle varie sorgenti luminose. Queste sono dei vettori composti da 4 float. I valori rappresentano le componenti RGB (rosso, verde, blu) del colore e la trasparenza (Alpha).

```
// intensità luminosa
float4 LightAmbientIntensity = { 0.8f, 0.8f, 0.8f, 1.0f };
float4 LightDiffuseColor     = { 2.0f, 0.0f, 0.0f, 1.0f };
```

La dichiarazione per il colore che verrà assegnato al materiale applicato alla mesh è molto simile:

```
// colore del materiale
float4 MaterialDiffuseColor = { 0.4f, 0.4f, 0.4f, 1.0f };
```

Concludono questa prima parte di inizializzazione le dichiarazioni delle varie componenti che servono all'elaborazione della scena 3D, come ad esempio la matrice di proiezione o la posizione della videocamera. Il tempo è associato alla variabile *Time* e, come vedremo, sarà molto utile in seguito, per l'implementazione delle animazioni.

```
// matrici di trasformazione
float4x3 World  : WORLD;
float4x3 View   : VIEW;
float4x4 ViewProjection : VIEWPROJECTION;
float3 CameraPos : CAMERAPOSITION;
// tempo corrente in secondi
float Time : TIME;
```

LO SHADER

Lo *shader* vero e proprio viene programmato nella funzione *VS()* che restituisce una struttura contenente la posizione del vertice elaborato *(Pos)* e il colore risultante dall'illuminazione diffusa *(Diff)*. La struttura è definita così:

```
// struttura per l'output del vertex shader
struct VS_OUTPUT
{   float4 Pos  : POSITION;
    float4 Diff : COLOR0; };
```

La funzione *VS()* è invece strutturata nel seguente modo:

MESH E TUTORIAL

È possibile trovare diverse mesh per la modifica e l'elaborazione dell'esempio proposto scaricando il Microsoft DirectX 9.0 SDK. Una volta installato il pacchetto sarà possibile dare un'occhiata agli esempi e ai tutorial presenti nella cartella Microsoft DirectX 9.0 *SDK\Samples\C++\Direct 3D*. Qui è possibile trovare mesh in formato *.x* nonché diversi file *".fx"* con numerosi effetti di esempio da analizzare. Sotto Microsoft DirectX 9.0 *SDK\Utilities\Bin\x86* si trova invece l'eseguibile di EffectEdit.

```
//--------------------------------------------------------
// Vertex shader
//--------------------------------------------------------
VS_OUTPUT VS(
// Posizione dei vertici nello spazio del modello
float3 InPos  : POSITION,
// Normali dei vertici nello spazio del modello
float3 InNormal : NORMAL)
{  VS_OUTPUT Out = (VS_OUTPUT)0;
    // Trasformazione delle posizioni e delle normali
    float3 Pos = mul(float4(InPos, 1),
                        (float4x3)World); // posizione
    float3 Normal = normalize(mul(InNormal,
                        (float3x3)World)); // normali
    // proiezione della posizione
    Out.Pos  = mul(float4(Pos, 1), ViewProjection);
    // illuminazione diffusa
    Out.Diff += CalcDiffuse(Normal, -DirFromLight);
    return Out; }
```

Vengono inizializzati come argomenti le variabili *InPos* e *InNormal* con la posizione e la normale del vertice da elaborare. Queste informazioni vengono convertite in coordinate compatibili con quelle del "mondo" all'interno del quale è posizionato il vertice, tramite le funzioni *mul()* e *normalize()* di *HLSL*. Successivamente viene riempita la struttura *VS_OUTPUT* creata in precedenza convertendo la posizione del vertice in coordinate proiettive, cioè, in pratica, coordinate utili per il disegno a schermo del vertice stesso. Questo avviene tramite la *ViewProjection*.
Schematicamente il percorso che seguono le coordinate del vertice è questo:

1. All'inizio il vertice è in coordinate "locali", lette ad esempio dal file *".x"* che contiene la mesh.
2. La coordinate locali vengono convertite in coordinate "globali", cioè viene posizionato il vertice tenendo conto di tutti gli oggetti della scena 3D con le relative traslazioni, rotazioni ecc.
3. Le coordinate globali vengono trasformate in coordinate "proiettive" per essere poi disegnate a schermo.

Il calcolo dell'illuminazione del vertice viene fatto dalla funzione *CalcDiffuse()* che tiene conto della normale del vertice e della direzione della

NOTA

HLSL E CG.

Abbiamo visto che i linguaggi per la manipolazione degli shader sono HLSL e Cg. Ma cosa li differenzia tra loro? In realtà poco e niente! Cg e HLSL sono virtualmente lo stesso linguaggio, sviluppato in cooperazione tra Microsoft e NVidia. I nomi sono differenti soprattutto per motivi commerciali. HLSL è parte delle DirectX e compila solo per queste API, mentre Cg può compilare anche per OpenGL. Di contro Cg è molto più indicato per le schede NVidia piuttosto che per le ATI.

DIVERSI TIPI DI ILLUMINAZIONE

```
float4 CalcHemisphere(float3 Normal, float3
                        DirToSky, float Occ)
{
    float4 Hemi = MaterialAmbientIntensity;
    // Fattore di occlusione
    Hemi *= (1 - Occ);
    // Calcola il lerp factor
    float LerpFactor = (dot(Normal, DirToSky) + 1) / 2;
```

1 **Una particolare tecnica di illuminazione realistica consiste nel calcolare la porzione emisferica di illuminazione di ogni singolo vertice. È possibile inserire nel nostro shader questo codice.**

```
    // Interpolazione tra il colore del cielo e quello del
                                              terreno
    Hemi *= lerp(GroundColor, SkyColor, LerpFactor);

    return Hemi;
}
```

2 **Più "nascosto" (occluso) sarà il vertice dalla geometria della mesh, minore luce riceverà e quindi sarà più scuro quando verrà disegnato. Questo codice completa la funzione *CalcHemisphere()***

```
float4 CalcSpecular(float3 Normal, float3
                  DirFromLight, float3 EyeToVertex)
{
    float3 R = normalize(reflect(DirFromLight,
                                        Normal));
    return MaterialSpecularColor * LightSpecularColor
            * pow(max(0, dot(R, -EyeToVertex)),
                              MaterialSpecularPower/4);
}
```

3 **Un altro tipo di calcolo della luminosità del vertice è quello "speculare". Questa tecnica utilizza la posizione dell'osservatore (EyeToVertex) per stabilire se la luce si rifletterà completamente sul vertice rendendolo molto chiaro, "a specchio".**

4 **Il risultato delle varie tecniche applicate può arrivare a essere molto realistico se visto in movimento. Ricordiamo inoltre che generalmente, la presenza di texture migliora di molto la resa visiva di una mesh.**

luce. Il corpo funzione di *CalcDiffuse()* è il seguente:

```
//-----------------------------------------------------------
// Calcola la luminosità diffusa
//-----------------------------------------------------------
float4 CalcDiffuse(float3 Normal, float3 DirToLight)
{  return MaterialDiffuseColor * LightDiffuseColor
                        * max(0, dot(Normal, DirToLight)); }
```

Il colore finale associato al vertice è il prodotto tra *MaterialDiffuseColor, LightDiffuseColor* (definiti inizialmente) e il valore del prodotto scalare tra i vettori passati come argomento. Senza scendere in spiegazioni articolate ci basti sapere che questo valore è tanto più alto quanto le direzioni dei vettori operandi sono simili. L'effetto che si ottiene è un colore molto chiaro quando il prodotto scalare è alto e la luce "batte" direttamente sul vertice. Il colore assegnato è scuro quando il prodotto scalare è negativo: in questo caso l'espressione *max(0, dot(Normal, DirToLight))* vale *0* e l'intera moltiplicazione assume valore nullo. Completa il codice di questo effetto la dichiarazione della "tecnica" *(technique)* che utilizza la funzione *VS()* appena definita. Una technique si definisce analogamente a come segue:

```
//-----------------------------------------------------------
// Techniche per il disegno a schermo del modello
//-----------------------------------------------------------
technique Draw
{   pass P0
    { VertexShader = compile vs_2_0 VS();} }
```

Si usa, come si può vedere, la clausola "compile" seguita dalla versione di *vertex shader* che si utilizza (in questo caso la 2.0) e il nome della funzione che elabora il vertice. Viene definito anche il nome del "passo" di elaborazione *("pass p0").* Ci possono essere uno o più passi in uno shader e si può selezionarli singolarmente o nell'insieme tramite le funzionalità offerte da *EffectEdit.*

ANIMARE IL BIPLANO

Così com'è il nostro shader non fa nulla (di utile). Illumina il vertice mediante un algoritmo noto di diffuse e effettua le trasformazioni "standard" viste in precedenza: *locale-globale-proiettivo.* In pratica abbiamo acquistato una Ferrari per tenerla in garage! Liberiamo quindi la potenza dei VS con un esempio semplice quanto significativo. Aggiungiamo un effetto "ondulatorio" al modello rappresentato. Per farlo aggiungiamo innanzitutto la seguente riga di codice nel corpo funzione di *VS():*

```
// Modifica la posizione dei vertici
Pos += CalcVertexAnimation(InPos);
```

Questa istruzione somma al vettore posizione *Pos* il vettore che è risultato della funzione *CalcVertexAnimation(),* definita come segue:

```
//-----------------------------------------------------------
// Animazioni da applicare ai vertici del modello 3D
//-----------------------------------------------------------
float3 CalcVertexAnimation(float3 Offset)
{ return float3( sin(Time + Offset.x), sin(3*Time +
                                  Offset.x), 0) * 0.2; }
```

Viene restituito da questa funzione un vettore ottenuto manipolando il vettore-parametro di ingresso con la funzione matematica seno *(sin())* utilizzata congiuntamente alla variabile *Time* definita precedentemente, per ottenere l'effetto "ondulatorio". Giocando con le variabili *Time* e *Offset* si possono ottenere in maniera molto semplice svariati effetti. Si provino ad inserire, ad esempio, le seguenti istruzioni in alternativa:

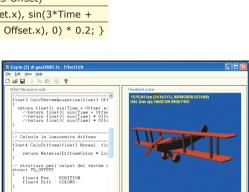



Fig. 1: ***Il nostro esempio in funzione all'interno del programma EffectEdit***

```
return float3( sin(Time + Offset.x), sin(Time +
                                  Offset.y), 0);
return float3( sin(Time) + Offset.x, sin(Time) +
                                  Offset.y, 0);
return float3( sin(Time) * Offset.x, sin(Time) *
                                  Offset.y, 0);
```

Gli effetti che si otterranno saranno per la maggior parte strani e difficilmente prevedibili a priori. Ecco un motivo per apprezzare un tool di sviluppo veloce come *EffectEdit.*

CONCLUSIONI

In questo articolo abbiamo visto come creare un semplice Vertex Shader utilizzando un linguaggio di manipolazione di alto livello. Abbiamo inoltre visto come sviluppare in maniera rapida un effetto, visualizzandone immediatamente il comportamento tramite il tool *EffectEdit.* Con un po' di creatività e la giusta esperienza il lettore riuscirà sicuramente ad ottenere svariati tipi di effetti semplicemente modificando di poco il codice di esempio.

Alfredo Marroccelli

Due trucchi per migliorare VB.NET

Con List View e Tree View, controlli molto potenti e comodi. Creiamo delle classi personalizzate che ci consentono di estenderne le funzionalità secondo le nostre esigenze

Ora, la maggioranza di voi, conoscerà il controllo "ListView". Lo scopo dell'esistenza di questo controllo è quello di mostrare all'utente dei dati in forma organizzata. Ad esempio: si può utilizzare per rappresentare un elenco tale che ciascun elemento dell'elenco sia composto di un'icona e un'etichetta; oppure si può visualizzare in maniera tale che vengano mostrati dei dettagli, ad esempio: Nome, Cognome, Data di Nascita.
Le modalità con cui la ListView mostra i dati che contiene, sono profondamente influenzati dalla sua property "View". Ad esempio in modalità "List" ciascun elemento appare composto di un'etichetta e una piccola icona alla sua destra. In modalità "Details" ciascun elemento appare su una linea separata e ciascuna linea contiene oltre al nome dell'elemento anche dei dettagli che vengono espressi nella proprietà "SubItems". Il punto è che per esempio in modalità "Details" solo la prima colonna della ListView è editabile da parte dell'utente. Perciò, se per esempio volessimo utilizzare questo controllo per consentire di inserire dei dati, la cosa ci costerebbe non pochi problemi. In questo articolo estenderemo le capacità di una ListView, al fine di renderla editabile in ogni sua parte da chi utilizza il nostro programma. Aggiungeremo inoltre qualche funzionalità di ricerca, e infine effettueremo lo stesso tipo di modifica anche per i controlli di tipo "TreeView".

ESTENDERE LA CLASSE BASE

La programmazione ad oggetti permette di ereditare le caratteristiche di un oggetto base per poterlo "specializzare" secondo le proprie necessità. Grazie al fatto che .NET è una libreria ad oggetti è possibile applicare questo concetto a quasi tutte le sue classi. In questo caso la classe in esame è la ListView che mette a disposizione proprietà, metodi ed eventi per gestire una list view. Il nostro scopo è estendere questa di modo che abbia il seguente comportamento:

Fig. 1: *L'effetto che vogliamo ottenere è quello di rendere editabili i singoli campi di una listview*

- Quando l'utente effettua un doppio click su una cella appartenente a una qualunque colonna, deve apparire un campo di testo, editabile e in un primo momento contenente il valore della cella selezionata. Ovviamente, il campo di testo deve avere le stesse dimensioni e la stessa posizione della cella.

- La pressione del tasto invio sulla cella di testo deve confermare l'immissione del dato e di conseguenza sostituire il valore del campo di testo con quello della cella selezionata. La pressione del tasto esc o il un click del mouse al di fuori del campo di testo non deve apportare modifiche alla ListView.

- Al termine della fase di editing, il campo di testo deve essere nascosto, e il controllo torna alla ListView

Vediamo, a livello di codice, quali sono i passi per specializzare la classe in questione. Prima di tutto occorre dichiarare il suo nome e da quale classe base ereditare:

```
public class ListViewEnhanced : ListView
```

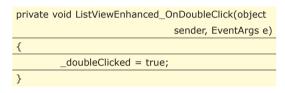
Se la classe definisse dei metodi virtuali, sarebbe necessario sovrascriverli per specificarne il comportamento se invocati. Nel caso specifico, la classe ListView non dispone di entità di questo tipo, per cui nessun passo extra sarà necessario per la specializzazione.
Cominciamo a definire delle variabili private, visibili solo all'interno della classe. Utilizzeremo queste variabili come struttura contenente i dati per definire la posizione e i contenuti della casella di testo che ci servirà per gestire l'editing della cella selezionata.

```
private TextBox _textbox;
private bool _doubleClicked;
private ListViewItem _item;
private int         _subItemIndex;
```

Utilizzeremo un valore booleano per sapere se l'utente ha effettuato un doppio click, un ListViewItem per tenere traccia dell'oggetto selezionato all'interno della list view e, infine, un intero per memorizzare l'indice della colonna selezionata.

```
public ListViewEnhanced()
{
    _textbox = new TextBox();
    _textbox.Visible = false;
    _textbox.BorderStyle = BorderStyle.FixedSingle;
    _textbox.KeyPress += new
    KeyPressEventHandler(_textbox_OnKeyPress);
    this.Controls.Add(_textbox);

    _doubleClicked = false;

    base.FullRowSelect = true;
    base.View = View.Details;
    base.AllowColumnReorder = false;
    base.LabelEdit = false;

    this.DoubleClick += new
     EventHandler(this.ListViewEnhanced_OnDoubleClick);
    this.MouseUp += new
    MouseEventHandler( ListViewEnhanced_OnMouseUp);
    this.ColumnClick += new
    ColumnClickEventHandler(
                    ListViewEnhanced_OnColumnClick);
}
```

All'interno del costruttore, ovvero del metodo chiamato automaticamente al momento della creazione dell'oggetto, inseriamo tutto il codice necessario ad inizializzare la classe. Prima di tutto creiamo un oggetto di tipo text box specificando alcune caratteristiche come il bordo non tridimensionale, l'invisibilità e la funzione che risponderà all'evento KeyPress. Successivamente forziamo il controllo ListView di base ad assumere alcune caratteristiche come la selezione dell'intera riga al posto della sola prima colonna, la vista di tipo “details”, ad evitare che sia possibile spostare l'ordine delle colonne (cosa che comporterebbe molto lavoro extra per ricavarne il vero indice) ed, infine, la disabilitazione dell'editing standard. Ultima operazione necessaria è quella di definire tre funzioni che rispondano agli eventi di doppio click, rilascio del tasto del mouse e click sulla testata della colonna.

```
private void ListViewEnhanced_OnDoubleClick(object
                                    sender, EventArgs e)
{
            _doubleClicked = true;
}
```

All'interno della funzione che risponde all'evento del doppio click non faremo altro che impostare il booleano a true.

```
private void ListViewEnhanced_OnMouseUp(
                          object sender, MouseEventArgs e)
{
   if (_doubleClicked)
   {
      _item = GetItemAt(e.X, e.Y);
```

Il cuore della classe risiede all'interno del gestore dell'evento MouseUp che si verifica quando un pulsante del mouse viene rilasciato. Come accennato all'inizio, dobbiamo differenziare il comportamento del controllo; quando avviene un doppio click l'utente vuole editare il contenuto della cella, quando il click è singolo l'utente vuole nascondere il campo di testo. Questa differenza può essere controllata analizzando il valore del booleano impostato durante l'evento DoubleClick. Nel caso in cui questo sia stato impostato a true, il codice chiama il metodo GetItemAt() esposto dalla classe ListView per ricavare l'elemento selezionato in base alle coordinate del puntatore del mouse. L'evento MouseUp fornisce un parametro di tipo MouseEventArgs che, tra le altre cose, contiene proprio le coordinate del puntatore del mouse al momento del rilascio del pulsante.

```
if (_item != null)
{
   Rectangle lviBounds;
   int      subItemX;
   lviBounds = _item.GetBounds(
```

```
                        ItemBoundsPortion.Entire);
```

Innanzitutto occorre controllare che il doppio click sia stato effettuato su una riga della ListView e non, magari, su una zona del controllo priva di elementi. Questo si effettua facilmente controllando che il ritorno del metodo GetItemAt() non sia uguale a null. A questo punto con un elemento selezionato valido possiamo passare alla fase di calcolo delle sue dimensioni. Con GetBounds() è possibile ricavare un oggetto Rectangle contenente le dimensioni del rettangolo che descrive la riga selezionata.

```
subItemX = lviBounds.Left;
for (int i=0; i<this.Columns.Count; i++)
{
```

A questo punto comincia l'algoritmo di calcolo dell'indice della colonna a cui appartiene la cella selezionata. La coordinata della prima colonna corrisponde alla coordinata sinistra del rettangolo; poi comincia il ciclo per calcolare l'indice della colonna selezionata.

```
ColumnHeader h = this.Columns[i];
if (e.X < subItemX+h.Width)
{
   _subItemIndex = h.Index;
   Rectangle textRect = new Rectangle(subItemX,
                  lviBounds.Top, this.Columns[i].Width,
                                       lviBounds.Height);

   _textbox.Bounds = textRect;
   _textbox.Text = _item.SubItems[
                               _subItemIndex].Text;
   _textbox.Visible = true;
   _textbox.BringToFront();
   _textbox.Focus();

   break;
}
subItemX += h.Width;
```

Per ricavare la colonna selezionata l'algoritmo somma l'ampiezza della colonna ricavata tramite l'indice del ciclo for con quella memorizzata all'interno della variabile di appoggio subItemX. Se la coordinata X del puntatore del mouse al momento del click è maggiore di questo valore la variabile di appoggio sarà incrementata con il valore dell'ampiezza della colonna. Si passerà alla successiva colonna fino a che non si troverà un valore superiore alla coordinata X del puntatore del mouse (a meno che il doppio click è stato effettuato su una riga ma al di fuori di tutte le colonne). A questo punto non rimane che memorizzare l'indice della colonna selezionata, calcolare la dimensione del rettangolo della cella e impostare il campo di testo ancora nascosto con queste dimensioni. Infine, il campo di testo sarà reso visibile e mostrerà il testo contenuto nella cella. Il break all'interno del ciclo è fondamentale. Immaginiamo, infatti, di aver selezionato la cella della prima colonna. La condizione dell'if sarebbe sempre verificata anche per ogni altra colonna con un conseguente malfunzionamento del controllo.

```
private void _textbox_OnKeyPress(
                  object sender, KeyPressEventArgs e)
{
   switch(e.KeyChar)
   {
      case (char)Keys.Escape:
          _textbox.Visible = false;
      break;
          case (char)Keys.Enter:
          if (_subItemIndex == 0)
          {
             _item.Text = _textbox.Text;
          }
          else
          {
             _item.SubItems[_subItemIndex].Text =
                                     _textbox.Text;
          }
          _textbox.Visible = false;
          break;
      }
}
```

COME UTILIZZARE LA LISTVIEWENHANCED?

Dopo tutto questo lavoro sarebbe impensabile che l'utilizzo di questo controllo comporti delle difficoltà. Ed infatti la cosa è molto semplice; utilizzando Visual Studio .NET occorre seguire questi passi:

1. **Aggiungere un riferimento all'assembly che contiene la classe ListViewEnhanced scegliendo il menu Project | Add Reference...**
2. **Trascinare una list view sopra la windows form**
3. **Andare nel codice della windows form e cambiare ogni riferimento alla classe ListView con la nuova classe ListViewEnhanced**

La variabile di tipo ListViewEnhanced può essere utilizzata come una classica list view e riempita con i valori opportuni

ANNULLARE L'EDITING

L'ultima funzione da analizzare è quella che gestisce alcuni tasti speciali che corrispondono a specifiche azioni come quelle di annullare l'editing o di completarlo. Se il tasto esc

viene premuto durante la fase di editing tutto ciò che è stato scritto viene ignorato e la casella di testo viene nascosta. Se, invece, il tasto invio viene premuto durante la fase di editing, il programma controlla l'indice della colonna della relativa cella selezionata; se è la prima colonna, il codice prende il testo direttamente dall'oggetto ListViewItem, altrimenti utilizza la collezione SubItems per accedere ai suoi sotto elementi.

RICERCA RAPIDA DI UN ELEMENTO DELLA TREE VIEW

Immaginiamo di aver riempito una tree view, con una classica struttura ad albero formata da genitori e figli. L'applicazione potrebbe implementare una ricerca fornendo la possibilità all'utente di selezionare una voce tra i risultati. La selezione della voce in questione dovrebbe spostare il cursore alla corrispondente voce nella TreeView Come fare?
Verrebbe istintivo pensare ad un ciclo tra tutti i nodi della tree view fino a trovare quello che ha lo stesso valore di quello selezionato. Ma esiste un modo più elegante, funzionale e rapido: l'utilizzo di una HashTable.
L'HashTable è una classe .NET che permette di gestire una zona di memoria dove memorizzare una tabella formata da due colonne: chiave e valore. Entrambe le colonne possono contenere degli oggetti generici object in modo da poter memorizzare qualsiasi cosa, da interi a stringhe fino a classi intere.
Nel caso della tree view si può utilizzare una HashTable per memorizzare nella colonna valore un oggetto di tipo TreeNode identico a quello che si sta inserendo all'interno della tree view. Come chiave si deve utilizzare un valore univoco che ci possa permettere, poi, di rintracciare facilmente questo valore nella tabella.
Immaginiamo di dover realizzare un tool che visualizzi le caratteristiche di un file contenuto all'interno di una TreeView. Quest'ultima sarebbe formata da un'alberatura simile a quella di Gestione Risorse di Windows, dove il nodo principale è rappresentato dalla lettera associata all'unità del disco e poi i vari rami si diffondono per raffigurare l'alberatura del disco. In questo caso la chiave della nostra tabella potrebbe essere il nome del file compreso di percorso. Questo garantirebbe una univocità della voce contenuta nella tabella.

```
Hashtable  _treenodes = new Hashtable();
```

```
TreeNode root = tvRoot.Nodes[0];
root.Tag = "C:";
_treenodes.Add(root.Tag, root);
```

Queste poche righe di codice permettono di mantenere allineata la tabella in memoria con la vista ad albero. Durante l'aggiunta di un nodo possiamo utilizzare un'informazione univoca come chiave della tabella (in questo caso il valore memorizzato nella proprietà Tag) e inseririre come valore il nodo stesso. In questo modo ricavare l'oggetto all'interno della tabella e utilizzarlo per la selezione diventerà veramente semplice:

```
TreeNode node = (TreeNode)_treenodes[
                              lbResults.SelectedItem];
tvRoot.SelectedNode = node;
tvRoot.SelectedNode.Expand();
tvRoot.Focus();
```

L'elemento selezionato da una list box, dopo aver effettuato una ricerca, è utilizzato come chiave per ricavare il valore corrispondente nella tabella Hash in memoria.

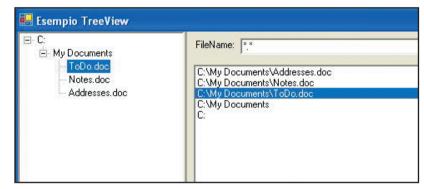



Fig. 1: *La Tree view deve essere mantenuta allineata con i dati ricercati*

Questo valore, poi, viene dato in pasto alla tree view per selezionare il nodo e provocare l'apertura di tutta l'alberatura fino alla sua selezione.

CONCLUSIONI

La potenza della programmazione ad oggetti risiede proprio nella possibilità di estendere la funzionalità delle classi base senza dover riscrivere il codice a partire da zero. nel nostro caso è evidente quanto abbiamo sfruttato la OOP creando due comodi controlli.
Si tratta di due piccoli trucchi che però rendono sensibilmente la vita più semplice al programmatore, laddove l'interfaccia utente deve essere adattata ad esigenze specifiche.

Fabio Claudio Ferracchiati

Java e JDBC per l'accesso ai dati

Alla scoperta delle tecniche per utilizzare i database senza problemi Poche righe di codice per ottenere un accesso universale ai vari server senza doversi preoccupare delle loro differenze

REQUISITI

Conoscenze richieste
Principi di Java

Software
J2SDK, Eclipse

Impegno

Tempo di realizzazione

La comunicazione fra una generica applicazione e il suo database è sempre stata, un grosso problema, alcuni l'hanno addirittura simpaticamente definita: "La torre di Babele tecnologica". Effettivamente spesso ci siamo trovati di fronte a database che comunicavano con la nostra applicazione solo ed esclusivamente utilizzando linguaggi proprietari, con l'immaginabile conseguenza che cambiare database significava anche cambiare il linguaggio di comunicazione. Una fatica decisamente onerosa anche per i programmatori più diligenti, figuratevi per noi programmatori Java che siamo abituati alla filosofia: "Scrivo una volta, compilo il codice e il tutto funziona ovunque.." Nella storia SQL ha rappresentato sicuramente un primo passo nella facilitazione all'accesso ai database, ma Java JDBC API credo che possa essere definito un vero e proprio traguardo. Infatti SUN con questa tecnologia, ha coniato un vero è proprio standard per far comunicare la stessa applicazione con un'ampia varietà di database impiegando semplicemente il linguaggio SQL.

COS'È JDBC E A COSA SERVE

Una prima attenzione va sicuramente dedicata al nome: "JDBC", secondo SUN non è assolutamente l'acronimo di JavaDataBaseConnectivity ma semplicemente un marchio registrato composto da quattro lettere unite fra loro in maniera del tutto casuale. Il tutto sarebbe sicuramente credibile se il suo storico antagonista, Microsoft, non avesse adottato il nome "ODBC" OpenDataBaseConnectivity per definire la sua tecnologia standard per l'accesso ai database. Insomma una "J" al posto della "O" una casualità decisamente sottile per non essere notata.

JDBC è un'API, Application Program Interface, quindi un insieme di classi e interfacce Java che consen-

Fig. 1: Come aprire il progetto nel workspace

COSA SERVE PER COMINCIARE

- **Scaricare l'ambiente di sviluppo eclipse all'indirizzo: www.eclipse.org vi consiglio di utilizzare la versione 3.0.3. Ricordo che eclipse NON va installato, per utilizzarlo è sufficiente scaricarlo e lanciare il comando *eclipse.exe***
- **Scaricare il server per il database mySql all'indirizzo: www.mysql.com/ Scompattare il file *mysql-4.1.10a-win32.zip* e installato mediante il *Setup.exe***
- **Scaricare l'interfaccia grafica per gestire il database all'indirizzo: http://dev.mysql.com/downloads/other/mysqlcc.html Scompattare il file *mysqlcc-0.9.4-win32.zip* e installato mediante il Setup.exe**
- **Scaricare i driver JDBC per mySql all'indirizzo: http://dev.mysql.com/downloads/connector/j/3.1.html Scompattare il file *mysql-connector-java-3.1.7.zip* NON richiede alcuna installazione.**
- **Copiare la cartella del progetto *Agenda,* disponibile sul CD della rivista, sotto il vostro Workspace: .... *eclipse\workspace\[Agenda]***
- **Aprire il progetto nel vostro workspace come mostrato in Figura 1**
- **Portare fra le librerie del progetto il: *mysql-connector-java-3.1.6-bin.jar* come mostrato in Figura 3**
- **lanciare l'applicazione *Agenda* seguendo le istruzioni di Figura 2**

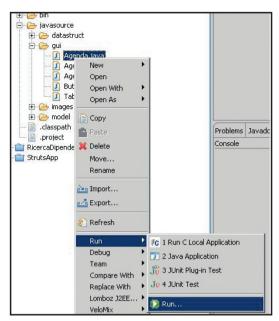
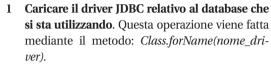



Fig. 2: ***Come lanciare l'applicazione Agenda***

tono l'interazione fra programmi Java e database; essendo operante a livello delle chiamate, un programma può accedervi utilizzando semplicemente metodi o chiamate a funzioni. In pratica la sua funzione principale è quella di consentire ad uno sviluppatore Java di connettersi ad un database, inviare richieste SQL, recuperare e gestire i risultati. Sul mercato sono disponibili centinaia di driver JDBC, almeno uno per ogni database comunemente utilizzato (MySql, Oracle, Access, ecc..). La sua vera grande potenza è quella di offrire un'interfaccia standard per tutti i tipi di database; questo significa che una query JDBC funzionante su MySql potrebbe tranquillamente funzionare anche su Oracle. Perdonatemi, ma in questo caso il condizionale è d'obbligo ed è riferito soprattutto alle problematiche che possono insorgere a causa delle divergenze che esistono fra i nomi dei tipi di dato dei diversi database. Tranquilli… questa problematica non rappresenta comunque una limitazione di JDBC, poiché può essere risolta direttamente dal programma utilizzando i metadati forniti dalla connessione JDBC. È importante sapere che esiste anche un driver JDBC denominato Bridge JDBC- ODBC.
Questo, utilizzando ODBC come intermediario, consente si utilizzare all'interno di JDBC anche i driver ODBC.

Fig. 3: ***Come importare del progetto il: mysql-connector-java-3.1.6-bin.jar***

COME FUNZIONA JDBC

Lavorare con JDBC è veramente semplicissimo. Le operazioni necessarie possono essere riassunte in quattro passi. Di seguito mostreremo solo la "teoria" ma più avanti, per ogni passo, ne vedremo anche un esempio pratico.

1 **Caricare il driver JDBC relativo al database che si sta utilizzando**. Questa operazione viene fatta mediante il metodo: *Class.forName(nome_driver)*.
nome_driver - rappresenta, come si può facilmente intuire, il nome del driver che si vuole caricare e varia per ogni driver.
2 **Aprire la connessione con il database utilizzando il driver precedentemente caricato.** *java.sql.DriverManager.getConnection(ConnectionUrl);*
Questa operazione viene svolta mediante una chiamata al metodo *getConnection()* delle classe *DriverManager*.
ConnectionUrl – rappresenta l'url per la connessione, indica sia il tipo di driver che si vuole utilizzare sia dove andare a recuperare il database.
3 **Eseguire le istruzioni SQL**. Attivata la connessione, la si può utilizzare per create oggetti Statement mediante i quali, possono essere creati comandi SQL.
4 **Estrarre i risultati restituiti dalle istruzioni SQL**. Questa operazione si rende possibile utilizzando l'interfaccia *ResultSet* che mette a disposizione dei metodi per elaborare le righe restituite ed estrarre i valori di ogni singola colonna.

LE CLASSI PER LA CONNESSIONE

Alcune di queste classi non ci serviranno mai, ma è utile ugualmente conoscerle per ottenere una visione d'insieme dei meccanismi che regolano JDBC

- **java.sql.Driver -** A meno che voi non vogliate scrivervi un vostro driver JDBC, non utilizzerete ma direttamente questa interfaccia, poiché i produttori di driver JDBC lo hanno già fatto per voi. Essa rappresenta semplicemente lo start iniziale per ottenere una connessione con un database.
- **java.sql.DriverManager** - Al contrario della precedente, è una classe e non un'interfaccia. Il suo compito è quello di mantenere la lista dei driver registrati per una particolare applicazione. Ogni applicazione può registrare uno o più driver JDBC diversi fra loro, la classe *DriverManager*, per mezzo del metodo *getConnection()*, può "decidere", in base all'url passato come argomento, quale driver utilizza-

ESEMPIO DI QUERY PARAMETRICA

Le query parametriche vengono utilizzate in tutti quei casi in cui l'interrogazione di una tabella coinvolge l'uso di un parametro, un esempio è il seguente:

```
public static String
            updateUtente()
{ StringBuffer query =
        new StringBuffer();
   query.append(
            "UPDATE ");
   query.append("agenda
     .anagraficautenti");
   query.append(
      "SET indirizzo = ?,
         cap = ? [......]);
   query.append(
          "WHERE ");
   query.append(
          "nome = ? AND
       cognome = ? AND
     dataNascita = ?");
   return query.toString();
}
```

query.append("SET indirizzo = ? [......]) ? **questa istruzione indica che l'attributo "indirizzo" è un parametro della query. prima dell'esecuzione ho la facoltà di sostituire questi parametri con i valori reali.**

```
prepStatment.setString
     (7, updateUtente
        .getNome()) ?
```

questa istruzione indica che al settimo parametro della query voglio assegnare il valore *updateUtente .getNome();*. Infine l'esecuzione della query non avviene più passando la Stringa al metodo *executeUpdate()* dell'interfaccia, ma avviene richiamando direttamente il metodo sull'interfaccia stessa:

```
prepStatment.execute
            Update();
```

re, fra quelli registrati, per creare una nuova connessione con il database. La classe *DriverManager* fornisce due metodi: *registerDriver()* e *deregisterDriver()* che consentono ad un driver di registrarsi o di de-registrarsi dalla lista.

- **java.sql.Connection** - L'interfaccia *Connection* rappresenta la vera connessione logica con il database. Il metodo *getConnection()* di *DriverManager* restituisce un oggetti di tipo *Connection* che, grazie ad una serie di metodi, consente di svolgere moltissime operazioni, fra le quali: preparazione ed invio di statement SQL, gestione delle transazioni, gestione degli eventuali commit o rollback delle operazioni SQL eseguite.

LE CLASSI PER L'ESECUZIONE SQL

Statement, PreparedStatement e **ResultSet** rappresentano alcune delle interfacce fondamental del core JDBC

- **java.sql.Statement** - Gli oggetti di tipo *Statement*, consentono di eseguire semplici query SQL dove per semplici s'intende query che non fanno uso di parametri. La classe fornisce diversi metodi ma i più utilizzati sono sicuramente:
- **executeQuery(String sqlString)** questo metodo, prende come argomento una stringa SQL e restituisce un oggetto di tipo *ResultSet. ExecuteQuery(..)* può essere utilizzato per tutte le chiamate SQL dalle quali ci si aspetta un valore di ritorno, tipicamente dei dati prelevati da un database.
- **executeUpdate(String sqlString)** ha lo stesso funzionamento dell'*executeQuery(…)*, l'unica differenza sta nel fatto che invece di restituire un

JDBC IN PRATICA

Per gli esempi pratici di questo articolo, ho deciso di implementare una semplice applicazione che permette di gestire i contatti di una Agenda. Per i soliti motivi spazio temporali non vi spiegherò nel dettaglio tutta l'applicazione, ma solo le parti inerenti a JDBC, come al solito, i sorgenti dell'applicazione sono disponibili nel cd della rivista. Un buon editor Java è Eclipse.

INSTANZIARE IL DRIVER

```
public class LoadDriver
{
   public static void getDriver()
   {
      try
      {
         Class.forName("com.mysql.jdbc.Driver"
                          ).newInstance();
      }
      catch(Exception ex)
      {…}
   }
}
```

1 ***mysql.jdbc.Driver*** **rappresenta il nome del Driver JDBC utilizzato dal database *mySql.* "Fisicamente" il driver è contenuto all'interno del jar: *mysql-connector-java-3.1.6-bin.jar* importato nel nostro progetto.**

LA CONNESSIONE

```
private Connection getConnection()
{
   Connection connection = null;
   Try
   {
      connection=(Connection)java.sql
            .DriverManager.getConnection(
              jdbc:mysql:// 88.777.222.00
          /agenda?user=root&password=);
   }
   catch(SQLException ex)
   {…} return connection;
}
```

2 **Questo metodo restituisce una connessione attiva con il database "*agenda*" dell'applicazione *Agenda*, viene quindi richiamato ogni volta che abbiamo la necessità di interagite con il database.**

I COMANDI SQL

```
public boolean insertNuovoUtente(
                   DataUtenteStruct utente)
{ Statement statment = null;
   Connection con = getConnection();
   try
   { …
      String query = QuerySQL.insertUtente(
                                  utente);
      statment = con.createStatement();
      statment.executeUpdate(query);
      … }
   catch(SQLException ex)
   {…}
}
```

3 **Inseriamo un nuovo utente nel database. Entra in gioco un'altra classe molto importante per la nostra applicazione ed è la: *model.QuerySQL*, all'interno di quest'ultima vengono implementate tutte le query.**

LA SINTASSI DI GETCONNECTION

jdbc:mysql ?
Identifica quale driver JDBC utilizzare.

//88.777.222.00 ?
Identifica l'indirizzo della macchina dove gira il server, quello che vedete è l'indirizzo di una macchina test
/agenda?user=root&password= ?
Identifica il nome del database al quale connettersi (*agenda*), seguito dall'autenticazione dell'utente che si sta connettendo (*user* e *password*). In Pratica:
"Connettiti al database posto sotto la macchina //88.777.222.00 con il driver jdbc:mysql con l'utente root, la password non è specificata"

oggetto di tipo *ResultSet*, restituisce il numero di righe per le quali ha effettuato l'operazione di update. Un tipico utilizzo di questo metodo è per eseguire query di *UPDATE* e di *INSERT*.

- **execute(String sqlString)** viene utilizzato in quei casi in cui non si sa se eseguire un executeQuery(….) o updateQuery(….), questo tipicamente avviene quando lo Statement SQL viene creato dinamicamente, se l'esecuzione della query restituisce delle righe allora il metodo restituisce true, altrimenti restituisce false. Infine, con il metodo getResultSet() è possibile prelevare tutte le righe restituite dalla query.
- **java.sql.PreparedStatement** - Un oggetto di tipo *PreparedStatement* consente di scrivere delle query con parametri di input, l'interfaccia fornisce dei metodi in grado di sostituire tali parametri con dei valori reali. Essa estende l'interfaccia *Statement* e di conseguenza ne eredita tutte le funzionalità
- **java.sql.ResultSet** - Un oggetto *ResultSet*, rappresentail risultato di una query. Esso può essere schematizzato in forma tabellare dove sulle righe troviamo gli oggetti selezionati dal database, mente sulle colonne gli attributi di questi oggetti. L'interfaccia fornisce tutta una serie di metodi in grado di estrarre ogni singolo attributo per ogni singolo oggetto *(riga)*.

```
ResultSet rs = statment.executeQuery(query);
```

I metodi per estrarre i singoli attributi possono essere schematizzati nel seguente modo:

```
rs.get Type(int | String)
```

dove l'argomento può rappresentare o il numero o il nome della colonna, sempre facendo riferimento ai nomi e ai numeri presenti sul database, mentre il suffisso *type* rappresenta in quale tipi dato (*String, int,* ecc..) mi aspetto che venga restituito il risultato. Tutto quello che avete letto fin qui è illustrato in sei semplici passi nel box a fondo articolo

Valentina Muraglia

BIBLIOTECA

- **DATABASE PROGRAMMING WHITH JDBC AND JAVA (O'Reilly)**
- **JAVA DATABASE E PROGRAMMAZIONE CLIENT/SERVER (Apogeo)**

DEFINIAMO LA QUERY

```
public static String insertUtente(
                    DataUtenteStruct utente)
{
  StringBuffer query = new StringBuffer();
  query.append("INSERT INTO ");
  query.append("agenda.anagraficautenti");
  query.append( "(nome, cognome, [......]);
  query.append( "VALUES ('");
  query.append(utente.getNome() + "', \n");
  query.append( "'" );
  query.append(utente.getCognome()
                              + "', \n");
  ...
  query.append( ")" );
  return query.toString();
}
```

4 **Molto semplicemente abbiamo creato una stringa contenente i comandi SQL che ci servono per manipolare il database. Si tratta di un metodo comodo che ci consente di passare questi comandi alla classe *model.query* che abbiamo visto in precedenza.**

QUERY PARAMETRICHE

```
public boolean update(
          DataUtenteStruct updateUtente)
{ Connection con = getConnection();
  PreparedStatement prepStatment = null;
  try
  { String query = QuerySQL.updateUtente();
    prepStatment = (PreparedStatement
           )con.prepareStatement(query);
    prepStatment.setString(
         1, updateUtente.getIndirizzo());
    ...
    prepStatment.setString(
           7, updateUtente.getNome());
    prepStatment.setString(
         8, updateUtente.getCognome());
    ...
    prepStatment.executeUpdate();}
catch(SQLException ex)
{...}
}
```

5 **Molto simile al suo quasi omonimo Statement, ci consente di eseguire delle query parametriche. La differenza sta nel fatto che la query non viene più costruita direttamente con i valori reali, bensì con parametri identificati dai vari '?'**

I RISULTATI

```
public List searchUtenti(
                  DataUtenteStruct utente)
{
  ResultSet resultSet = null;
  resultSet = (ResultSet)
           statment.executeQuery(query);
  while(resultSet.next())
  {
    DataUtenteStruct singoloUser = new
                      DataUtenteStruct();
    singoloUser.setNome(
           resultSet.getString("nome"));
    singoloUser.setNome(
              resultSet.getString(1));
    singoloUser.setCognome(
        resultSet.getString("cognome"));
    ...
  }
}
```

6 **Il *ResultSet* rappresenta l'oggetto che "fisicamente" ci consente di estrapolare il risultato di una query. Un suo utilizzo completo è visibile nel metodo *searchUtenti(...)* sempre nella nostra classe: *model.BcAgendaModel()***

Quel chiacchierone di Visual Basic

Se in altri ambienti realizzare applicazioni che usano la sintesi vocale può sembrare particolarmente complesso, in VB questa tecnica è ormai consolidata. In pochi passi si realizza un'applicazione completa

Ho iniziato ad usare la SAPI *(Speech Application Programming Interface)* per sviluppare un'applicazione dedicata ai non vedenti che potevano, in tal modo, scrivere i propri documenti e sentirli leggere dal PC. Naturalmente questo, per quanto sia lodevole, è soltanto una delle numerose applicazioni in cui potrebbe avere senso utilizzare un motore di sintesi e riconoscimento vocale.
Sta a voi individuare i campi d'applicazione.

IL KIT DI SVILUPPO

Per sviluppare un'applicazione di "sintesi vocale" abbiamo bisogno del kit Microsoft Speech SDK contenente:

- le SAPI *(Microsoft Win32-compatible speech application programming interface);*
- un motore per il riconoscimento vocale *(Microsoft continuous speech recognition engine);*
- un motore per la sintesi vocale *(Microsoft concatenated speech synthesis engine)* conosciuto anche come *text-to-speech (TTS) engine.*

Il kit fornisce inoltre alcuni strumenti di sviluppo per la verifica e il test dello sviluppo, alcuni esempi e la documentazione sulle principali caratteristiche dell'SDK.
Per sviluppare la nostra applicazione abbiamo usato la versione SDK contenente la SAPI 4.0 che supporta l'automazione OLE. Questo fa sì che non solo i linguaggi come C/C++ possono usare le SAPI, ma anche quei linguaggi come Visual Basic, C# e javascript capaci di accedere agli oggetti di automazione.

COME FUNZIONANO LE API NEL TTS

Il programma fa uso del motore delle SAPI per eseguire una sintesi vocale *(text-to-speech)* che consiste nel passaggio tra *testo->formato binario->parlato.* In particolare la nostra applica-

Conoscenze richieste
Basi di C#, Excel

Software
Visual basic 6.0, Speech API SDK

Impegno
Tempo di realizzazione
COME INIZIARE

Affinché il nostro pc parli, è necessario scaricare ed installare il kit SDK 4.0 direttamente dall'indirizzo http://download.microsoft.com/download/speechSDK/Install/4.0a/WIN98/EN-US/SAPI4SDK.exe, oppure possiamo scaricare manualmente i singoli componenti che ci interessano. In particolare i files da installare sono:

- **spchapi.exe: questo file, scaricabile dal seguente indirizzo http://download.microsoft.com/download/speechSDK/Install/4.0a/WIN98/EN-US/spchapi.EXE, contiente le SAPI che fanno interagire l'applicazione con il motore TTS.**
- **Msagent.exe: questo file, che possiamo scaricare all'indirizzo http://activex.microsoft.com/activex/controls/agent2/MSagent.exe, permette di usare il componente Microsoft Agent 2.0 che utilizzeremo all'interno della nostra applicazione. Gli utenti di WINDOWS XP/2000 /Me possono saltare questo passo dato che i componenti del Microsoft Agent sono già installati per questi sistemi operativi.**

Se facciamo parlare il nostro PC noteremo un forte accento inglese, questo perché la piattaforma è progettata con la lingua inglese di default. Se vogliamo far parlare il nostro agente con un accento italiano dobbiamo scaricare due files del TTS:

- **lhttsiti.exe: che troviamo all'indirizzo http://activex.microsoft.com/activex/controls/agent2/lhttsiti.exe**
- **AgtX0410.exe: che troviamo all'indirizzo http://activex.microsoft.com/activex/controls/agent2/AgtX0410.exe**

Esistono diversi agent character (alcuni di questi sono il genio o il cane) che oltre a farci ascoltare il testo, lo scrivono all'interno di un "balloon". Ne possiamo scaricare diversi, in formato .acs dall'indirizzo www.msagentring.org/chars.htm

zione controlla le attività di sintesi attraverso l'interfaccia *ISpVoice Component Object Model (COM)* con cui prima crea un oggetto *ISpVoice* e successivamente richiama il metodo *ISpVoice::Speak* per tradurre vocalmente un qualsiasi testo.
Il metodo *ISpVoice::Speak* opera in due modalità:

- **Sincrona:** Termina solo quando l'applicazione ha completamente finito di parlare;
- **Asincrona:** Termina immediatamente e parla come se fosse un processo di background.

In modalità asincrona, se scriviamo del testo nuovo oltre quello che già avevamo, possiamo avere due effetti: in un caso, il parlato può essere interrotto e ascolteremo il testo appena inserito; nell'altro caso il testo sarà automaticamente accodato a quello che stiamo già ascoltando.

UN ESEMPIO DI REALIZZAZIONE

Realizzare un'applicazione per la sintesi vocale non è certamente un'impresa difficile, la particolarità consiste nel sviluppare funzionalità avanzate per la gestione del testo. Analizziamo il codice usato per realizzare le basi della nostra applicazione, successivamente mostreremo come renderla più completa. Per usare il motore TTS dobbiamo inserire nel nostro progetto il componente *Microsoft Agent Control 2.0.* Fatto ciò dichiariamo come oggetti globali all'interno della nostra form un attore, che nell'esempio è il mago merlino, ed una costante che definisce il percorso del folder dove si trova il file *merlin.acs*.

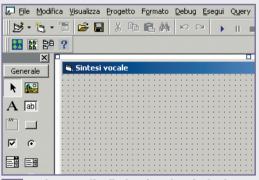
```
Dim merlin As IagentCtlCharacterEx
Const DATAPATH = "merlin.acs"
```

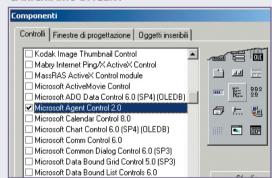
Nel caso in cui stessimo lavorando con WindowsXP, la costante *DATAPATH* corrisponde a

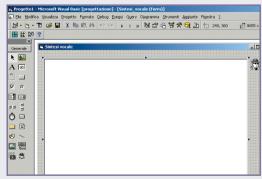
I TUOI APPUNTI

Utilizza questo spazio per le tue annotazioni

ECCO COME PARLA IL NOSTRO COMPUTER

CREIAMO UN NUOVO PROGETTO

1 **Dal menu File di Visual Basic selezioniamo Nuovo Progetto e scegliamo il tipo EXE Standard. Cambiamo il nome alla Form1 in Sintesi_vocale e diamo alla caption il valore Sintesi Vocale.**

CARICHIAMO L'AGENT

2 **Dal menu Progetto selezioniamo la voce Componenti e clicchiamo su Microsoft Agent Control 2.0. Comparirà l'icona dell'agente nella Casella degli strumenti. Quindi clicchiamo sull'icona dell'agent e importiamola all'interno della form.**

INSERIAMO LA CASELLA DI TESTO

3 **Inseriamo la casella di testo in cui andremo a scrivere e la chiamiamo Text1. Siamo pronti per scrivere il codice della nostra applicazione.**

PROVIAMO L'APPLICAZIONE

4 **Nella versione più semplice, la nostra applicazione apparirà con questo layout. Come vediamo l'attore è il mago ma è possibile sceglierne anche altri.**

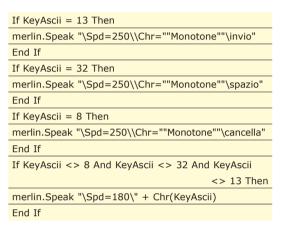
c:\WINDOWS\Msagent\char, in questa cartella saranno copiati i personaggi che traducono in parole il testo scritto. Più avanti vedremo che è possibile creare dei nuovi personaggi che dovranno essere salvati all'interno di questa cartella per poter essere utilizzati. Nella *Form_ Load()* carichiamo il personaggio seguendo il *DATAPATH* specificato prima ed assegnamo alla variabile *merlin* le proprietà di questo attore. Con la stringa *merlin.Balloon.Style = False* nascondiamo il balloon in cui compare il testo che scriviamo nella casella di testo, mentre con *merlin.LanguageID = &H410* specifichiamo che il mago Merlino dovrà parlare in italiano. L'ultima stringa ci permette di visualizzare il mago, contemporaneamente apparirà un'icona nella system tray.

```
Private Sub Form_Load()
    Agent1.Characters.Load "merlin", DATAPATH
   Set merlin = Agent1.Characters("merlin")
    merlin.Balloon.Style = False
    merlin.LanguageID = &H410
    merlin.Show
End Sub
```

Far parlare la nostra applicazione è veramente un gioco da ragazzi, basta infatti associare all'evento di pressione di un tasto l'azione *merlin.Speak*.

```
Private Sub Text1_KeyPress(KeyAscii As Integer)
merlin.Speak "\Spd=180\" + Chr(KeyAscii)
End Sub
```

All'interno della procedura abbiamo specificato la velocità con cui deve parlare il nostro agente vocale *(Spd=180)* e il carattere che deve pronunciare *(Chr(KeyAscii))*. Quando digiteremo del testo ascolteremo i caratteri corrispondenti ai tasti che stiamo premendo. Bisogna dire però il codice proposto è incompleto. Infatti se premiamo la barra spazio o il tasto invio, oppure il tasto cancella, non otterremo alcun risultato: il nostro agente non parla. Possiamo risolvere questo piccolo inconveniente con una semplice operazione. Sempre all'interno della procedura *Text1_KeyPress(KeyAscii As Integer)* definiamo una serie di controlli *If-Then* con i quali verifichiamo se il tasto premuto corrisponde a *barra spazio, invio* o *cancella.*

```
If KeyAscii = 13 Then
merlin.Speak "\Spd=250\\Chr=""Monotone""\invio"
End If
If KeyAscii = 32 Then
merlin.Speak "\Spd=250\\Chr=""Monotone""\spazio"
End If
If KeyAscii = 8 Then
merlin.Speak "\Spd=250\\Chr=""Monotone""\cancella"
End If
If KeyAscii <> 8 And KeyAscii <> 32 And KeyAscii
                                          <> 13 Then
merlin.Speak "\Spd=180\" + Chr(KeyAscii)
End If
```

Per ogni caso specifichiamo la velocità di pronuncia, il timbro e la parola associata al tasto che stiamo digitando. A questo punto vogliamo ascoltare tutto quello che abbiamo scritto nella casella di testo. Per far questo basta associare ad un tasto di funzione (per esempio *F1*) la proprietà *merlin.Speak Text1.Text* ed agganciare questa funzionalità all'evento *Text1_KeyUp*.

```
Private Sub Text1_KeyUp(KeyCode As Integer,
                                   Shift As Integer)
If KeyCode = 112 Then  '112 corrisponde al tasto F1
    If Text1.Text <> "" Then
       merlin.Speak Text1.Text
    End If
End If
```

Possiamo pensare di sviluppare altre funzionalità che ci fanno ascoltare il testo con la punteggiatura, gli spazi, i tag di invio ecc. che troviamo all'interno del testo. Queste informazioni sono sicuramente a supporto di un utente non vedente.

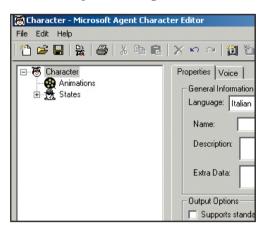
UNA VERSIONE PIÙ COMPLESSA

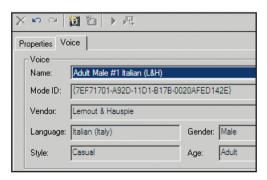
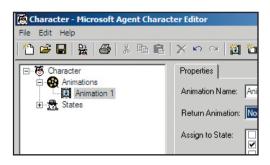
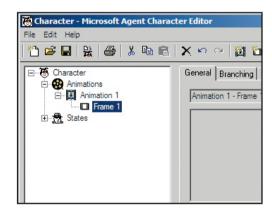
5 **Possiamo realizzare qualche funzione che agisce sul testo più che sul parlato. Con poche righe di codice l'utente sceglie il tipo, lo stile e la dimensione dei caratteri. Si può notare come non sia visibile l'attore, ma non è nascosto: è invisibile.**

AGENT CHARACTER EDITOR

Anche se l'editor non è compreso nel SDK, è comunque uno strumento su cui spendere due parole dato che è utile perché possiamo creare i

nostri attori preferiti con le caratteristiche grafiche e vocali più adeguate alla nostra applicazione. Nel caso specifico mi occorreva un character invisibile sullo schermo e nella *system tray*. Sono riuscito a realizzarlo con questo semplice editor che dà, comunque, la possibilità di associare numerose azioni al nostro personaggio. Ma vediamo come creare un personaggio invisibile così come lo desideravo.

1 Per iniziare dobbiamo scaricare l'editor direttamente dall'indirizzo *http://www.microsoft.com/msagent/downloads/developer.asp* ed installarlo sul nostro pc. Una volta avviato, il programma si presenta con la schermata riportata in **Figura 1**.

2 Eliminiamo l'opzione *Use word balloon* e spuntiamo la voce *Use synthesized speech for voice output*. Così facendo il testo non sarà stampato a video ma sarà parlato dal nostro personaggio.

3 Come riportato in **Figura 2**, sfogliamo la sezione *Voice* e selezioniamo dalla list box *Name* il tipo di voce che ascolteremo; possiamo scegliere una timbro maschile o femminile.

4 Creiamo una nuova animazione ciccando sull'icona, si aprirà la finestra rappresentata in **Figura 3**. Nella box *Assign to state* spuntiamo le proprietà *Hiding* e *Speaking*; il nostro personaggio sarà nascosto ma parlerà. Di conseguenza gli *States Hiding* e *Speaking* nelle proprietà generali del nostro character saranno associate all'*Animation1* che abbiamo appena creato.

5 Adesso realizziamo un *New Animation Frame* cliccando sull'icona, si aprirà la finestra rappresentata in **Figura 4**. Nella finestra *General* aggiungiamo un'immagine in formato *.bmp* che è l'unico supportato. Per avere un personaggio invisibile, bisogna prima creare un'immagine trasparente che associeremo in questo momento.

6 Dopo aver assegnato l'immagine possiamo creare il nostro personaggio selezionando dalle proprietà *File* il comando *Build Character*. Infine decideremo di salvare il nostro file nella cartella *c:\WINDOWS\msagent\char\character1.acs*.

CONCLUSIONI

Nonostante l'applicazione sia semplice da realizzare il potenziale di utilizzo è molto elevato specie se rivolta ad un particolare target di utenti. Possiamo arricchire il nostro programma con funzionalità elevate che danno valore aggiunto alla sintesi vocale. In questa versione ho usato le SDK 4.0 ma la Microsoft distribuisce anche la versione più aggiornata SDK 5.1.

Egidio Lancione

Attenzione a quell'uncino

Ci avventuriamo nei meandri del sistema operativo, per conoscere meccanismi che regolano la gestione degli eventi. Sapevate di poter modificare il comportamento del mouse a vostro piacimento?

Conoscenze richieste
conoscenze elevate di C#

Software
NET framework SDK, Visual Studio .NET, Windows 98/ME/2000/XP/2003

Impegno

Tempo di realizzazione

Riveliamo subito che lo scopo di questo articolo è scrivere un'applicazione di contachilometri per il mouse. Ci pensate a quanti chilometri percorre ogni giorno? Sono curioso di sapere quanti e mi sono messo in testa di scrivere questa applicazione. Ma, mentre scrivevo, mi sono reso conto che la cosa non è per niente banale. Il movimento del mouse è infatti un evento del sistema operativo. Cioè ogni volta che il mouse viene mosso, questo evento viene comunicato al sistema, che lo gestisce. Normalmente, ad esempio, la freccetta che indica il puntatore del mouse si muove sullo schermo. Perciò la mia prima idea è stata intercettare l'evento scatenato dal movimento del mouse e gestirlo al posto del sistema. E già questo non è stato un compito proprio facile. A questo punto mi sono anche reso conto che se mi sostituivo al sistema. Avrei dovuto gestire una procedura che contasse i chilometri percorsi dal mouse ma anche una procedura che spostasse il puntatore del mouse sullo schermo al posto mio. Il che è decisamente noioso. Vogliamo provare ad estendere il problema? Pensiamo ad un'applicazione che conti il numero di volte che un tasto viene premuto sulla tastiera. Anche qui ci sarebbe da intercettare l'evento e poi gestirlo, e gestire anche i compiti classici di sistema. Esiste un modo semplice per creare applicazioni di questo tipo? La risposta è si! Si chiama *"Gestione degli Hook di sistema"* ed è quello di cui parleremo in questo articolo.

COS'È UN HOOK

Un *hook* è un meccanismo del sistema operativo mediante il quale è possibile intercettare particolari eventi di sistema, senza per questo impedire che essi giungano alle applicazioni che normalmente li gestiscono. Ad esempio è possibile intercettare movimenti e click del mouse, o anche le pressioni dei tasti, o ancora eventi come l'apertura di una finestra, il suo ridimensionamento, e chi più ne ha più ne metta. È possibile, associare dunque delle funzioni di callback, invocate automaticamente quando un evento ben definito si verifica, e dunque intraprendere particolari azioni.

CATENE DI HOOK

In realtà, ogni qualvolta il sistema rivela un evento, lo intercetta tramite un *Hook*. A questo punto viene dato il via a una sequenza di azioni che gestiscono l'evento. Questa sequenza di azioni prende il nome di *"Catema di Hook".* Perciò ad esempio premendo un tasto sulla tastiera, questo evento viene rilevato dal sistema e il controllo passa alla catena di Hook che lo gestisce. Va da sé che se voglio contare quanti chilometri percorre il mouse, devo registrare un *Hook* nella *catena di Hook* che gestisce l'evento dello spostamento del mouse. In questo modo il mio *hook* gestirà una procedura che memorizza lo spazio percorso, e appena questa procedura avrà termine il controllo passerà all'*hook* successivo.

Ciascun evento di sistema viene gestito da una catena di hook separata. La catena di hook prende il nome in inglese di *Hook Chain.*

LE MANI NEL CODICE

Per sfruttare gli hook è dunque necessario creare una hook procedure ed inserirla nella hook chain relativa. Una procedura di hook deve rispettare la seguente firma:

```
LRESULT CALLBACK HookProc(int nCode,
                    WPARAM wParam, LPARAM lParam);
```

Il nome della procedura, in questo caso *HookProc,* è naturalmente definibile a piacimento dallo sviluppatore. Il valore intero *nCode,* è un codice che determina l'azione da svolgere, e che dipende dal tipo di hook, cioè per ogni tipo esistono dei possibili valori

che *nCode* può assumere. Gli argomenti *wParam* ed *lParam* contengono dei valori che dipendono ancora dal tipo di hook, ma in genere contengono informazioni aggiuntive su un messaggio. Per installare una hook procedure in una *hook chain,* ed al contrario per rimuoverla da essa, l'API di Windows fornisce due funzioni, *SetWindowsHookEx* ed *UnhookWindowsHookEx.* La prima funzione installa una *hook procedure* in testa alla relativa *hook chain.*
Quando si verifica un evento che è associato ad un particolare tipo di hook, il sistema operativo inizia ad invocare le *hook procedure* della catena, ed all'interno di essa si determina se l'evento debba essere passato alla procedura successiva. In caso affermativo, l'evento viene propagato per mezzo della funzione *CallNextHookEx,*. Per alcuni tipi di hook è possibile solo monitorare gli eventi, in questi casi sarà il sistema operativo ad invocare la funzione *CallNextHookEx,* anche se all'interno della procedura non viene fatto esplicitamente. Ogni tipo di hook consente di monitorare un differente aspetto del sistema a messaggi di Windows.

UNA CLASSE .NET PER GLI HOOK

Il framework .NET non fornisce alcuna funzionalità interna per la gestione degli hook, e dunque è necessario ricorrere ai meccanismi di Platform Invoke, che consentono di utilizzare all'interno del codice managed le funzioni native del sistema operativo o comunque implementate in codice non gestito.
Mediante *P/Invoke* è dunque possibile continuare ad utilizzare le funzioni sopra introdotte, importandole all'interno del nostro codice gestito, come faremo ad esempio in C#. Le funzioni *SetWindowsHookEx, UnhookWindowsHookEx* e *CallNextHookEx* sono contenute all'interno della libreria di windows *user32.dll.* Per poterle utilizzare in C# dobbiamo innanzitutto dichiararle nella maniera seguente:

```
[DllImport("user32.dll")]
public static extern IntPtr SetWindowsHookEx(
                    HookType code, CbtHook.CbtProc func,
                              IntPtr hInstance,int threadID);
[DllImport("user32.dll")]
public static extern int UnhookWindowsHookEx(
                                          IntPtr hhook);
[DllImport("user32.dll")]
public static extern int CallNextHookEx(IntPtr hHook,
                  int nCode, Int32 wParam, IntPtr lParam);
```

L'uso dell'attributo *DllImport,* come potete notare, è in questo caso abbastanza semplice, basta indicare il nome della dll contenente la funzione che si vuole importare precedendo la dichiarazione stessa. Dichiarazione che avverrà con i modificatori *static* *extern,* ed in questo caso public. La dichiarazione del metodo deve naturalmente rispettare la firma della corrispondente funzione da importare, traducendo il tipo di ritorno e quelli dei parametri eventuali. Ad esempio la *SetWindowsHookEx,* restituisce un *handle HHOOK,* che in .NET è stato tradotto con il tipo *IntPtr. IntPtr* è infatti un tipo *struct* che rappresenta un puntatore o un *handle.*
Anche il parametro hInstance verrà dunque importato come un IntPtr. Il primo e l'ultimo parametro invece sono degli interi, e quindi è sufficiente utilizzare il tipo *int* di C#, anche se spesso, quando i valori ammissibili sono ben determinati, è possibile implementare dei tipi enumerativi in maniera da passare come argomento solo i valori corretti. Il primo parametro rappresenta infatti il tipo di hook che si desidera installare, ed esso può assumere uno dei valori che è possibile trovare nell'header file *WinUser.h* del Microsoft SDK. Con tali valori creeremo il tipo enumerativo *HookCode,* nel seguente modo:

```
public enum HookCode
{
  WH_JOURNALRECORD = 0,
  WH_JOURNALPLAYBACK = 1,
  WH_KEYBOARD = 2,
  WH_GETMESSAGE = 3,
  WH_CALLWNDPROC = 4,
  WH_CBT = 5,
  WH_SYSMSGFILTER = 6,
  WH_MOUSE = 7,
  WH_HARDWARE = 8,
  WH_DEBUG = 9,
  WH_SHELL = 10,
  WH_FOREGROUNDIDLE = 11,
  WH_CALLWNDPROCRET = 12,
  WH_KEYBOARD_LL = 13,
  WH_MOUSE_LL = 14
}
```

Con l'utilizzo di un tipo enumerativo, il codice diviene molto più semplice sia da scrivere, sia da leggere e mantenere. Ad esempio se vogliamo installare un hook per gli eventi relativi alla shell di Windows dovremo passare come argomento il valore 10, o corrispondentemente, in maniera più comoda il valore *HookCode.WH_SHELL.*
Nel nostro esempio gestiremo invece gli eventi di mouse e tastiera, in maniera da avere un log delle attività dell'utente sulle due periferiche, dunque utilizzeremo i valori 13 e 14.
Il secondo parametro della funzione è di tipo *HOOKPROC,* che rappresenta un puntatore ad una funzione di callback, che dovrà rispettare, come visto, la seguente firma:

```
LRESULT CALLBACK HookProc(
```

I TUOI APPUNTI

Utilizza questo spazio per le tue annotazioni

NOTA

IL PROGETTO

Sul CD o sul sito web di IoProgrammo trovate il codice completo dell'applicazione. Esso è costituito da una soluzione per Visual Studio 2003, ma naturalmente è possibile compilare da riga di comando per mezzo del compilatore c# (o lanciando il file compila.bat):

csc /out:hooks.exe Hook.cs KeyboardHook.cs MouseHook.cs Form1.cs

```
                         int code, WPARAM w, LPARAM l)
```

Corrispondentemente, in .NET, bisognerà implementare un delegate con la stessa firma della funzione di callback, traducendo in maniera opportuna i tipi dei parametri. In questo caso creeremo un delegate così:

```
public delegate int HookProc(int code, IntPtr wParam,
                                         IntPtr lParam);
```

Per chi non fosse a conoscenza del concetto di *delegate*, nel box laterale viene fornita una breve spiegazione.

LA CLASSE HOOK

È ora adesso di creare una classe in C#, in maniera da poter utilizzare le funzionalità fin qui esposte in maniera perfettamente object oriented, e traducendo il tutto sfruttando i concetti .NET di eventi e delegate. La classe che scriveremo sarà una classe astratta, in maniera da non poterne creare istanze, ma per poterla estendere in maniera semplice con classi che gestiscano ognuna un tipo di hook. Chiamiamo dunque questa classe base *Hook*, e dotiamola di un campo *m_nCode* di tipo *HookCode*, il cui scopo sarà quello di memorizzare il tipo di hook rappresentato da una istanza della classe, mentre il campo *m_hHook* conterrà l'handle alla procedura di hook una volta installata, o il valore 0 se l'installazione non va a buon fine.
Tale valore è esattamente quello restituito dalla funzione *SetWindowsHookEx*.
La classe *Hook* avrà dunque questa struttura:

```
public abstract class Hook
{
   [DllImport("user32.dll")]
   public static extern int SetWindowsHookEx(
                        HookCode hCode, HookProc lpfn,
                         IntPtr hInstance, int threadId);
   [DllImport("user32.dll")]
   public static extern bool UnhookWindowsHookEx(
                                            int hHook);
   [DllImport("user32.dll")]
   public static extern int CallNextHookEx(int hHook,
             int nCode, IntPtr wParam, IntPtr lParam);
   public delegate int HookProc(
             int nCode, IntPtr wParam, IntPtr lParam);
   protected int m_hHook;
   protected HookCode m_nCode;
   protected HookProc HookDelegate;
```

L'unico argomento del costruttore è un parametro del tipo enumerativo *HookCode* che identifica il tipo di hook da gestire:

```
   public Hook(HookCode hc)
   {
      m_code=hc;
   }
```

Implementiamo inoltre i due metodi *Install* ed *Uninstall*, che permettono di installare e disinstallare la procedura di hook rappresentata dal delegate *HookProc*. Essi non fanno altro che richiamare le funzioni per gli hook importate tramite *PInvoke*.

```
      public void Install()
      { m_hHook=SetWindowsHookEx(
          m_code,HookDelegate,Marshal.GetHINSTANCE(
                     Assembly.GetExecutingAssembly()
                            .GetModules()[0]), 0); }
      public void Uninstall()
      { if(m_hHook!=0)
        { bool bRet=UnhookWindowsHookEx(
                                          m_hHook);
          m_hHook=0;
          if(!bRet)
            throw new Exception("Impossibile
                           disinstallare l'hook");
        }
      }
  }
```

La classe *Hook* così implementata costituisce una base astratta per delle classi che gestiscano dei particolari tipi di evento. Non ci resta adesso che implementare le classi che gestiscano particolari tipi di hook, cioè quelli che ci interessano, e che saranno concrete e istanziabili in una nostra applicazione.

HOOK PER MOUSE E TASTIERA

Supponiamo di voler monitorare gli eventi di mouse e tastiera, quello che dobbiamo fare è estendere la classe Hook in maniera opportuna, sfruttando i metodi generali in essa definiti, e creando ad esempio una gerarchia come quella in **Figura 1.**
Cominciamo con l'implementare una classe *MouseHook*, nel cui costruttore viene direttamente impostato il tipo *HookCode.WH_MOUSE_LL* ed inoltre impostiamo anche il delegate che costituisce la *hook procedure* da invocare quando si verifica un evento relativo al mouse.

```
public MouseHook():base(HookCode.WH_MOUSE_LL)
{
  this.HookDelegate=new HookProc(MouseHookProc);
}
```

Inoltre specifichiamo un evento, cioè un delegate da invocare all'interno della *hook procedure* e che verrà

I DELEGATE

**I delegate di .NET consentono di incapsulare all'interno di un oggetto un riferimento ad un metodo, in maniera analoga al meccanismo dei puntatori a funzione caro agli sviluppatori C e C++. L'oggetto delegate potrà quindi essere usato per invocare il metodo in esso incapsulato.
A differenza dei puntatori a funzione i delegate sono type-safe, object oriented e sicuri.**

consumato da un eventuale gestore interessato all'evento stesso:

```
public event MouseEventHandler OnMouseActivity;
```

Il metodo *MouseHookProc* dovrà invece rispettare la firma del delegate *HookProc*. Al suo interno gestiremo il messaggio di Windows, ricavandone i parametri, con i quali costruiremo un oggetto *MouseEventArgs* da passare al delegate *OnMouseActivity*, cioè a tutti gli eventuali gestori dell'evento stesso.
Il parametro *wParam* indica il tipo di evento, in questo caso l'unico evento gestito è il movimento del mouse, *WM_MOUSEMOVE*. Ulteriori informazioni, ad esempio le coordinate del puntatore del mouse sono ricavate dal parametro *lParam* con cui viene riempita una struttura *MOUSEHOOKSTRUCT*,

```
struct MOUSEHOOKSTRUCT
{
   public Point pt;
   public IntPtr hwnd;
   public uint wHitTestCode;
   public IntPtr dwExtraInfo;
}
```

Una volta riempiti i campi della struttura, da essa poi costruiamo un oggetto *MouseEventArgs* da passare al delegate *OnMouseActivity:*

```
private int MouseHookProc(int nCode, IntPtr wParam,
                                      IntPtr lParam)
{ if(nCode>=0)
   { if(wParam.ToInt32()==WM_MOUSEMOVE)
      { MOUSEHOOKSTRUCT mhs =
         (MOUSEHOOKSTRUCT) Marshal.PtrToStructure(
                    lParam, typeof(MOUSEHOOKSTRUCT));
         MouseEventArgs e=new MouseEventArgs(
                   (MouseButtons)nCode, 0, mhs.pt.X,
                                      mhs.pt.Y, 0 );
         OnMouseActivity(this, e); }
   }
   return Hook.CallNextHookEx(
              m_hHook,nCode,wParam,lParam);
}
```

Notate che come ultima istruzione viene invocato il metodo *CallNextHookEx* ereditato dalla classe base *Hook*. In maniera perfettamente analoga implementiamo una classe *KeyboardHook*, per monitorare la pressione dei tasti ed effettuare ad esempio una sorta di log dell'attività della tastiera. Questa volta abbiamo bisogno di una struttura *KBDLLHOOKSTRUCT* siffatta:

```
struct KBDLLHOOKSTRUCT
{ public int vkCode;
   public int scanCode;
   public int flags;
   public int time;
   public IntPtr dwExtraInfo;
}
```

Mentre la procedura di gestione sarà implementata in questa maniera:

```
private int KeyboardHookProc(int nCode, IntPtr
                        wParam, IntPtr lParam)
{  if(nCode>=0)
   { KBDLLHOOKSTRUCT mhs = (KBDLLHOOKSTRUCT)
            Marshal.PtrToStructure(lParam,typeof(
                          KBDLLHOOKSTRUCT));
      KeyEventArgsEx e=new KeyEventArgsEx((Keys)mhs
                  .vkCode,wParam.ToInt32());
      OnKeyboardActivity(this, e); }
   Hook.CallNextHookEx(m_hHook,nCode,wParam,
                                   lParam);
   return 0;
}
```

In questo caso il parametro *wParam* indica il tipo di evento sui tasti, cioè se si tratta di una pressione o del rilascio del tasto. Invece *lParam* indica il codice del tasto premuto. La classe standard *KEyEventArgs* però consente di impostare solo il secondo. Dunque estendiamo quest'ultima aggiungendo al costruttore anche il tipo di evento che si è verificato:

```
public class KeyEventArgsEx:KeyEventArgs
{  public enum KeyMessage
   { KeyDown=256,
      KeyUp=257 }
   public KeyMessage Message;
```

Fig. 1: *La gerarchia derivata dalla classe Hook*

P/INVOKE

Platform Invoke o PInvoke è il meccanismo che permette al codice gestito di .NET di individuare le funzioni esportate da una DLL nativa, caricarle in memoria, ed invocarle effettuando una sorta di traduzione dei suoi argomenti.

```
public KeyEventArgsEx(Keys vkCode,int msg)
                                        :base(vkCode)
    { this.Message=(KeyMessage)msg;}
}
```

UN ESEMPIO APPLICATIVO

Con le classi *MouseHook* e *KeyboardHook* possiamo creare un'applicazione Windows Forms, per monitorare i due relativi tipi di eventi. Ad esempio possiamo contare la distanza percorsa dal mouse, come una specie di contachilometri, anzi contapixels, mentre con la seconda classe creeremo un log dei tasti premuti. Implementiamo dunque una Form con una label che mostri la distanza percorsa dal mouse ed una casella di testo in cui invece scrivere-mo il codice di ogni tasto premuto durante la vita dell'applicazione stessa. Prima di tutto però è necessario creare le istanze delle due classi *Hook* ed invocare il metodo *Install* per ognuna delle due:

```
mh=new MouseHook();
mh.OnMouseActivity+=new MouseEventHandler(
                                mh_OnMouseActivity);
mh.Install();
kh=new KeyboardHook();
kh.OnKeyboardActivity+=new KeyEventHandler(
                                kh_OnKeyboardActivity);
kh.Install();
```

Gli eventi relativi verranno gestiti tramite i due metodi *mh_OnMouseActivity* e *kh_OnKeyboardActivity*, che rispondono alla firma dei delegate rispettivi. Nel primo viene calcolata la distanza fra la posizione attuale del puntatore del mouse e quella memorizzata all'ultimo verificarsi dell'evento. Il risultato verrà memorizzato nel campo *totPixels* e mostrato come testo della label *labPixels:*

```
private void mh_OnMouseActivity(object sender,
                                        MouseEventArgs e)
{   Point ptNew=new Point(e.X,e.Y);
    totPixels+=CalcDist(ptPrec,ptNew);
    ptPrec=new Point(e.X,e.Y);
    this.labPixels.Text=String.Format("{0:D14}",
                                            totPixels);
}
```

Il metodo utilizza la seguente procedura che calcola la distanza fra due punti, i più ferrati in matematica non hanno bisogno in questo caso di alcuna spiegazione aggiuntiva:

```
private int CalcDist(Point p1,Point p2)
{   int dist=(int)Math.Sqrt(Math.Pow((p2.X-p1.X),2)
                            +Math.Pow((p2.Y-p1.Y),2));
    return dist;
}
```

Il metodo seguente invece ricava il codice del tasto premuto ed il tipo di evento, aggiungendo una nuova linea alla casella *rtfBox*:

```
private void kh_OnKeyboardActivity(object sender,
                                        KeyEventArgs e)
{ KeyEventArgsEx ex=(KeyEventArgsEx)e;
    this.rtfBox.AppendText(Environment.NewLine
        +ex.Message.ToString()+" key:"+ex.KeyCode);
}
```

Alla chiusura della applicazione, se i due hook sono stati installati, essi verranno rimossi semplicemente invocando il metodo *Uninstall*

```
if(mh!=null)
            mh.Uninstall();
if(kh!=null)
            kh.Uninstall();
```

In questa maniera abbiamo terminato la nostra semplice applicazione di esempio, la cui unica form è mostrata nella figura seguente:

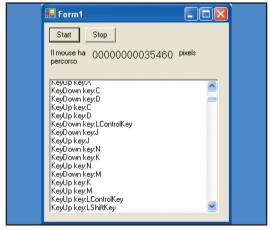



Fig. 2: *L'applicazione per monitorare mouse e tastiera*

CONCLUSIONI

Sebbene l'applicazione realizzata non abbia una vera e propria utilità pratica, essa costituisce un buon esempio di partenza per l'utilizzo degli hook di Windows, e soprattutto, l'articolo ha mostrato come gestire concetti un po' più avanzati all'interno di un'applicazione gestita, facendo uso della tecnologia Platform Invoke. Naturalmente l'argomento è molto vasto e complesso per essere esaurito in un solo articolo, ma il nostro scopo era quello di fornire un'introduzione e di stuzzicare il vostro appetito di conoscenze.

Antonio Pelleriti

BIBLIOTECA

- **PROFESSIONAL C#** *Robinson et al.* (Wrox Press)
- **PROGRAMMING WINDOWS WITH C#** *Petzold* (Microsoft Press)
- **APPLIED MICROSOFT .NET PROGRAMMING** *Richter* (Microsoft Press)

L'AUTORE

Potete contattare l'autore per suggerimenti, critiche o chiarimenti all'indirizzo e-mail antonio.pelleriti@ioprogrammo.it.

Una lista nera per gli spammer

Le Black List un metodo utilizzato a livello mondiale per combattere lo spam. Importante da conoscere se si gestisce un mail server, o semplicemente per saperne di più su che fine fa la nostra posta

In questo articolo analizzeremo il funzionamento delle *"Black List"*, un metodo utilizzato sovente lato server e invisibile all'utente finale, la cui conoscenza è però importantissima per chi gestisce un "mail server" o per chi ricevendo molta posta spazzatura desidera fare in modo che allo spammer venga negato l'utilizzo della rete per fini non legittimi. Dalle nostre parole avrete già capito che il metodo in questione non coinvolge il proprio computer e le proprie risorse personali, si riferisce invece a un metodo più generale che tende a combattere ad un livello più ampio lo spam. L'idea è molto semplice. Alcuni siti web raccolgono le segnalazioni degli utenti in relazione a "denunce" di spam. I server SMTP dei vari provider quando ricevono una mail interrogano questi siti web e, se il mittente del messaggio non è elencato nelle liste degli spammer, allora viene regolarmente inviato all'utente, viceversa viene bloccato. In sostanza, un utente non riceverà mai i messaggi di spam perché questi messaggi verranno filtrati a livello di server e non a livello utente. Allo stato attuale tutti i grandi provider come anche i piccoli, utilizzano questo metodo per combattere lo spam. Da Interbusiness, a Wind a Tiscali, ognuno utilizza in un modo o nell'altro servizi di *Black List* internazionali.

REQUISITI

Conoscenze richieste
Basi di Java

Software
J2SE.1.4

Impegno

Tempo di realizzazione

SEI SULLA LISTA NERA!

Perché una black list abbia effetto, è necessario che l'elenco dei mittenti sia il più possibile completo. Per ottenere un elenco completo ed esaustivo è imprescindibile una organizzazione a livello mondiale. Non ha senso infatti creare la propria lista personale sul proprio PC o in una rete locale. Per questo motivo sono nate molte organizzazioni che gestiscono un elenco globale di mittenti indesiderati. Nella maggioranza di questi non viene memorizzato il mittente, ad esempio *max2005@libero.it*, ma l'indirizzo IP del server che invia il messaggio. Il protocollo di posta di Internet (SMTP) funziona infatti attraverso la connessione diretta tra i diversi server.

Quando si invia un messaggio di posta, lo si trasmette tramite un server SMTP. Ad esempio, questo può essere *mail.libero.it.* Il server a cui corrisponde questo nome di dominio riceve il messaggio e lo manda al server di destinazione. Se ad esempio il messaggio è destinato a *lele2005@tiscalinet.it*, il server *mail.libero.it* cerca di connettersi al server di tiscali. L'indirizzo preciso del server viene restituito in un record *"MX"* attraverso una ricerca su DNS.

Se il server *mail.libero.it* è inserito in black list ed il server di posta di *tiscalinet.it* supporta la gestione della black list, il messaggio viene rifiutato. Questo avviene perché il server che invia è presente nelle black list. Ovviamente alcune volte succede che i mittenti siano inseriti in black list erroneamente.

L'utilizzo di una black list ha diversi vantaggi. Ad esempio i maggiori server di posta supportano questa feature in modo nativo. In questo modo l'introduzione di questo ulteriore elemento di sicurezza avviene semplicemente configurando il server. Inoltre la protezione è subito attiva, grazie al database presente nei siti che forniscono il servizio. Detto questo, è bene notare anche che è anche possibile comunicare ai fornitori della black list eventuali nuovi mittenti indesiderati, in modo che vengano inseriti nei database.

OPEN RELAY

Ci sono diverse cause per cui un mail server può finire in una black list. Il motivo principe per cui un sever può finire in Black List è quello di essere un *"Open relay"*. La definizione di *Open Relay* è piuttosto ampia. In linea del tutto generale è un *Open Relay* un server che consente ad utenti anonimi di

inviare la posta. Ad esempio, se il provider pippoepluto.it fornisse il servizio di posta e un qualunque utente riuscisse a inviare la posta tramite questo provider senza autenticarsi, questo sarebbe un *Open Relay.*
Chiaramente se un utente può fare spam senza poter essere identificato è potenzialmente pericoloso. Viceversa si pensa che gli utenti autenticati siano fidati e che non facciano spam. L'autenticazione al server SMTP avviene quando l'utente fornisce una username e una password per inviare la posta.
Attenzione a non confondere password e utente per l'invio con quello per la ricezione. Il protocollo che consente di inviare la posta è l'SMTP quello che consente di ricevere il POP3. Quando un server SMTP invia la posta, un altro server SMTP la riceve e la smista *(dispatch)* a un server POP3. L'utente destinatario si connette al server POP3 e "scarica" la posta tramite username e password. Un'altra protezione contro il relay è tipicamente quella di limitare l'invio della posta ai soli utenti che utilizzano un indirizzo IP sulla stessa rete su cui è posizionato il server SMTP. Avete mai provato a usare ad esempio il server di tiscali per inviare le email mentre siete collegati tramite un provider differente, ad esempio Telecom? Dubito che ci riuscirete.

INSIDIE DELLE BLACK LIST

Supponete di aver preso un bel virus. Supponete che questo virus non faccia altro che nascondere al suo interno un Mail Server. Uno spammer potrebbe usarvi come ponte per inviare la posta. Risultato: il vostro indirizzo IP potrebbe essere segnalato a una Black List; risultato secondario: l'intero provider proprietario di quell'indirizzo IP potrebbe essere "bannato" dalla rete internet. Attenzione, non è un'eventualità improbabile, ai provider di grosse dimensioni capita spesso e volentieri. Questi troiani sono utilizzati dai malintenzionati per inviare posta in modo anonimo, con un frequente cambio di IP del mittente. Proprio per evitare di essere intercettati da liste di blocco. Nella maggior parte dei casi è possibile essere rimossi dalle liste di blocco. Tipicamente è sufficiente segnalare alla Black List di avere rimosso la causa per cui si era stati inseriti in BL. Per fare questo è necessario contattare gli amministratori della lista in oggetto utilizzando le procedure da loro fornite. I gestori della BL controlleranno che la causa sia stata effettivamente rimossa, ad esempio proveranno a usarvi come Open Relay, e se verificano che ogni cosa è tornata al suo posto, vi elimineranno dalle Black List.

COME INTERROGARE LE BLACK LIST

Si noti che le liste di blocco non possono essere scaricate integralmente, perché si compongono di megabyte di informazioni. Alcuni fornitori di servizio prima lo prevedevano, ma sono stati costretti a limitare questa possibilità. L'accesso alle liste avviene invece per interrogazione. L'utente o il sistema digitano l'indirizzo IP sospetto ed il

RECORD MX

Un record MX è un tipo di informazione contenuto nel sistema di nomi dei domini (DNS) che indica come deve essere reindirizzata la posta Internet. Quando un messaggio viene inviato ad un server di posta, questo esegue una query DNS richiedendo il record MX del destinatario.

ALCUNI PROVIDER DI BLACK LIST

Su Internet sono disponibili diversi siti che forniscono implementazioni di black list. Alcuni di questi siti collaborano tra di loro mettendo in comune le proprie basi dati. Organizzazioni degne di nota in merito alla gestione di liste di blocco sono:

Spamhaus - SBL è un database in tempo reale di indirizzi IP di mittenti verificati di posta spazzatura. Per "mittenti" si intendono spammers, gruppi di spam e servizi di supporto agli spammer. È mantenuto dal gruppo di supporto del progetto Spamhaus. È fornito gratuitamente per aiutare gli amministratori della posta a gestire meglio i flussi di input. SBL è richiamabile in tempo reale dai sistemi di mail presenti in Internet. SBL permette agli amministratori di identificare e bloccare connessioni da indirizzi IP coinvolti nell'invio di posta non richiesta. Il database è aggiornato ogni ora nell'arco della settimana da un gruppo di supporto composto da statunitensi, inglesi, olandesi, italiani, giapponesi e serve diversi Paesi in tutto il mondo.

Abusive Hosts Blocking List (AHBL) - Questa organizzazione è gestita da una divisione del Summit Open Source Development Group (SOSDG), un'azienda di sviluppo di tecnologie open source. Si basa su informazioni ottenute da varie sorgenti su Internet e su strumenti di tracciatura e database proprietari. Anche AHBL fornisce un database di indirizzi di macchine coinvolte con la posta spazzatura. Il servizio più rilevante, tra i tanti offerti da AHBL, è il DNSbl ip4r. Questo consente la ricerca di host che inviano posta abusiva, o che contiene virus, oppure tentativi di phising.

Spamcop - Questa azienda mantiene la SpamCop Blocking List (SCBL), una lista di indirizzi utilizzati per inviare posta indesiderata agli utenti SpamCop. Il database può quindi essere utilizzato da fornitori di servizi ed utenti individuali per bloccare e filtrare posta spazzatura.

DSBL - Il gruppo *Distribuited Sender Blackhole List* si occupa di raccogliere indirizzi IP di computer che hanno inviato speciali mail di test alla stessa DSBL. In pratica, DSBL attende un messaggio sulla casella listme@listme.dsbl.org, quando questo arriva, l'indirizzo IP del mittente viene inserito in elenco. Infatti, un utente normale non ha motivo per inviare messaggi a quella particolare casella. Ma gli spammer che raccolgono in modo automatico indirizzi di posta dalle pagine Web non lo sanno. Dunque raccolgono anche questo indirizzo e cominciano ad inviare messaggi indesiderati. E questo li inserisce nell'elenco degli spammer gestito da DSBL.

sistema dice se è presente nel suo database. Ovviamente questi fornitori di servizio devono essere sostenuti da hardware molto potente e banda a volontà. Si pensi a tutte le interrogazioni che possono ricevere. Per averne una idea bisogna moltiplicare il numero di server di posta presenti in Internet per il numero di messaggi gestiti al giorno. Il risultato è sicuramente un numero considerevole.

SPAMMER E BLACK LIST

I malintenzionati utilizzano diverse tecniche per aggirare le liste di blocco e le normative contro la posta indesiderata. Ad esempio creano un falso *ISP (Internet Service Provider)* con il quale acquistano dai grandi provider, banda e connessioni di rete. Giustificando la richiesta, con il fatto che il falso ISP ha clienti a sua volta che si connettono in dial-up. Lo spammer in realtà poi utilizza questa connessione di rete non per fantomatici clienti, ma per inviare i propri messaggi spazzatura. Quando qualcuno si lamenta con il grande provider dello spam inviato con le sue connessioni, questo inoltra le lamentele all'ISP fantoccio. Queste lamentele dovrebbero essere dirette agli ipotetici utenti. Ma essendo tutto un imbroglio, ovviamente le lamentele cadono nel vuoto. Inoltre, grazie al fatto che gli spammer acquistano grandi quantità di banda, spesso i grandi ISP sono disposti a chiudere un occhio. Alcune volte la connettività è acquistata "off-shore", ad esempio in Cina. Questo viene fatto per evitare le autorità dove vigono leggi severe, come negli Stati Uniti.

Ma spesso anche gli host off-shore sono presto bloccati e terminati. Per questo lo spammer deve cambiare spesso provider, saltando da uno all'altro. Un trucco utilizzato spesso è quello di utilizzare lo stesso computer per ospitare siti di supporto per le truffe via Internet e per inviare la posta spazzatura. Viene poi utilizzato un server DNS che si aggiorna automaticamente per puntare ad un nuovo indirizzo IP quando cade la linea in dial-up oppure il provider taglia il collegamento. Questi sistemi sono supportati da una copia di backup sempre pronta. Quando le autorità tagliano il collegamento ad un computer, ne è già pronto un altro per sostituirlo. Secondo Spamhaus, l'80% della posta spazzatura è generata da solo 200 organizzazioni, per di più note. Queste organizzazioni utilizzano circa 500-600 persone, che cambiano continuamente alias e domini. Queste organizzazioni sono elencate nel *ROSKO (Register of Known Spam Operations)* e operano illegalmente. Per essere inseriti nel ROSKO, uno spammer deve essere stato oggetto di annullamento del contratto da almeno 3 fornitori di servizio, per violazioni relative alla posta spazzatura. Le sottoreti e gli indirizzi IP sotto il controllo dei soggetti elencati nel *ROSKO* sono inseriti automaticamente in alcune liste di blocco, come SBL.

UN OCCHIO AL CODICE

Per capire come può essere possibile eseguire una interrogazione viene preso in esame un server di posta specifico, James. Questo server scritto in Java è un server di posta SMTPT e POP3. Include anche il

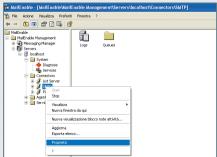
NOTA

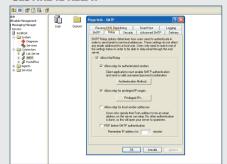
QUERY MX

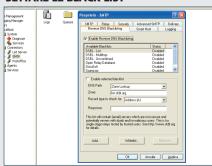
Il record MX contiene un elenco di host che accettano posta per quel dominio. Per ciascun host viene restituita anche la distanza. Il server di invio quindi sceglie il server più vicino e cerca di stabilire una connessione per l'invio del messaggio.

CONFIGURARE MAILENABLE CONTRO LO SPAM

Mailenable è un server di mail disponibile in versione Free all'indirizzo http://www.mailenable.com. La versione Free contiene tutte le caratteristiche della versione professional, eccetto alcuni servizi come ad esempio l'interfaccia di configurazione via web.

LE PROPRIETÀ SMTP

1 **Dall'interfaccia di gestione di Mailenable cliccare su connectors, poi su SMTP e con il tasto destro su Proprietà**

GESTIRE IL RELAY

2 **Selezionare la tabsheet Relay e settare i parametri per l'autenticazione SMTP. L'autenticazione integrata è una buona scelta**

SETTARE LE BLACK LIST

3 **Selezionare la tabsheet "Reverse Dns Blacklisting" ed abilitare la BL tramite l'apposito Flag. Abilitare poi le singole black list da utilizzare**

supporto al protocollo per le news NNTP. Il suo nome completo è Apache Java Enterprise Mail Server (*http://james.apache.org/*) ed è sviluppato dall'*Apache Foundation*. Inoltre, James è una piattaforma per la gestione della posta elettronica. Il gruppo di sviluppo ha infatti realizzato delle API per scrivere codice Java che elabora i messaggi di posta elettronica. Nel fare questo è stato creato il termine mailet, sulla falsariga dei nomi Applet, Servlet e MIDlet. Le mailet possono essere utilizzate ad esempio per generare risposte automatiche, oppure aggiornare un database, evitare la posta indesiderata, archiviare i messaggi o altre cose ancora. Per capire se un mailet deve elaborare uno specifico messaggio interviene un altro componente, il matcher. Le API di mailet e matcher sono sviluppate come parte del progetto James, mentre il server vero e proprio ne fornisce una implementazione di riferimento.
È importante notare come James, mailet e matcher non sono API standard di SUN. JavaMail, diversamente, è uno standard ufficiale. Ma il suo scopo è leggere le mail dal server. È dunque una tecnologia client e non server. Al momento della stesura di questo articolo l'ultima versione di James stabile disponibile è la 2.2.0. Guarderemo quindi nei suoi sorgenti per capire come è stato implementato il controllo tramite black list in James.
La classe che ci interessa è *org .apache.james.transport.matchers.InSpammerBlacklist*, che è una sottoclasse di *GenericMatcher.* L'intercettazione dei messaggi avviene quindi attraverso un matcher. Il metodo più importante in queste componenti è *match()*. Se il metodo torna null l'elaborazione prosegue, altrimenti vengono ritornati i destinatari del messaggio. Il parametro di tipo Mail identifica invece il messaggio da controllare. Nel listato seguente è presente l'implementazione del metodo *match()* presente nella classe *InSpammerBlacklist* del server di posta James. I passi eseguiti dal codice sono i seguenti:

1. l'indirizzo del server di invio, ottenuto con il metodo *getRemoteAddr()* viene rovesciato. Se ad esempio è 220.186.10.20 viene convertito in 20.10.186.220. questa operazione viene svolta attraverso un oggetto *StringTokenizer* che separa gli ottetti cercando il punto (.) separatore;
2. all'indirizzo ottenuto viene aggiunto il nome del prefisso per la lista di blocco in uso, p.e. *sbl.spamhaus.org.* nel caso dell'esempio precedente si avrebbe *20.10.186.220.sbl.spamhaus.org*;
3. attraverso la classe *DNSServer* viene verificata l'esistenza nei DNS dell'indirizzo cercato. La ricerca avviene tramite il metodo *getByName()*. La classe *DNSServer* è implementata in una libreria Java di accesso a DNS presente sul sito *www.xbill.org;*
4. se viene sollevata una *UnknownHostException* significa che l'indirizzo non esiste, dunque il mittente è sicuro ed il metodo può ritornare null;
5. se non si sollevano eccezioni, vengono restituiti tutti i destinatari del messaggio, ottenuti con la chiamata al metodo *getRecipients()* sull'oggetto *Mail* passato come parametro al metodo. Questo comunica a James che il messaggio è stato eliminato dal matcher di controllo della lista di blocco.

```
public Collection match(Mail mail) {
  String host = mail.getRemoteAddr();
  try {
    //le liste di blocco si aspettano gli ottetti di un
    //indirizzo ip in ordine rovesciato
    StringBuffer sb = new StringBuffer();
    StringTokenizer st = new StringTokenizer(host, "
                                        .", false);
    while (st.hasMoreTokens()) {
      sb.insert(0, st.nextToken() + ".");}
    //aggiunge il prefisso di rete della lista di blocco
    //ad esempio sbl.spamhaus.org
    sb.append(network);
    //ricerca tramite DNS
    org.apache.james.dnsserver.DNSServer
                         .getByName( sb.toString());
   //se l'indirizzo viene trovato vuol dire che è
                              presente in black list
   return mail.getRecipients();
  } catch (UnknownHostException uhe) {
   //in caso contrario l'indirizzo è "pulito"
   return null;}
}
```

(codice sotto licenza Apache License, Version 2.0)

L'indirizzo *sbl.spamhaus.org* non punta ad un vero server. Se si richiede questo indirizzo con un browser si ottiene un errore. In realtà l'indirizzo punta a zone DNS bilanciate per il carico. Ciascuna zona è infatti suddivisa in più sottozone, distribuite in tutto il mondo. Ed i vari server DNS sono connessi tra di loro con canali ad alte prestazioni.
Inoltre, i DNS di Spamhaus sono implementati per le prestazioni. Questi aspetti permettono di ottenere tempi di interrogazione molto bassi, nell'ordine di millisecondi.
Inoltre, i risultati vengono inseriti in una cache per un certo periodo, in modo tale che le successive interrogazioni siano ancora più veloci. Ne consegue che non esiste un effettivo rallentamento del server di posta se questo implementa un filtro contro la posta spazzatura basato su liste di blocco. Inoltre, i moderni server elaborano i messaggi in parallelo, dunque anche se esiste un minimo ritardo nell'elaborazione di un messaggio, questo non rallenta l'operazione complessiva.

Massimiliano Bigatti

SUL WEB

Gli indirizzi dei database di mittenti di posta spazzatura sono i seguenti:

The SpamHaus Project
http://www.spamhaus.org/

The Abusive Hosts Blocking List
http://www.ahbl.org/

SpamCop.net
http://www.spamcop.net/bl.shtml

Distributed Sender Blackhole List
http://dsbl.org/main.

I servizi offerti sono completamente gratuiti. Potete usufruirne sia se siete un gestore di mail server, sia se siete un utente comune che vuole segnalare uno spammer.

PER SAPERNE DI PIÙ

DNS

I siti che forniscono l'accesso ai database degli indirizzi IP sospetti supportano query sotto forma di risoluzione di nome DNS. Quest'ultima è la tecnologia che viene utilizzata su Internet per la risoluzione dei nomi. Tra i vantaggi, la forte distribuzione e ridondanza dei dati.

Intel C++ Compiler un mostro di velocità

C++ è il linguaggio principe per ottenere prestazioni e controllo della macchina. Il compilatore Intel genera un assembly talmente ottimizzato da ottenere prestazioni 100 volte superiori alla media

Molte aree della programmazione, come il calcolo scientifico, la manipolazione real-time dei segnali e delle immagini e l'Intelligenza Artificiale, hanno in comune la necessità di prestazioni molto elevate. In questi casi una corretta ottimizzazione del codice non è sufficiente a garantire la velocità richiesta; la programmazione in un linguaggio d'alto livello, per quanto comodo e flessibile come il C++, comincia ad andare stretta, e diventa obbligatorio dare attenzione a tutti quei passaggi invisibili, che avvengono al di là della sfera d'influenza del programmatore. Quest'ultimo è quindi costretto o a cambiare linguaggio (con tutti i problemi relativi alla programmazione diretta in assembler), o ad affidarsi alla delicata pratica del fine-tuning delle prestazioni, da ottenere mediante le direttive e le opzioni che il compilatore mette a sua disposizione. In quest'ultimo caso è facilmente immaginabile quanto la scelta del compilatore più adatto alle proprie esigenze giochi un ruolo tanto fondamentale quanto complesso. La maggior parte dei compilatori fornisce tutta una serie di funzionalità ormai considerate classiche e quasi "tradizionali": semplificare il codice ridondante, eliminare potenziali ambiguità dei riferimenti in memoria, e molte altre. Una simile compilazione produce codice generalmente aderente alle specifiche dell'IA-32, le quali offrono la garanzia di essere supportate dalla maggior parte dei processori per PC in commercio. Questo permette al programmatore di ottenere una buona compatibilità, ma al prezzo di sacrificare il proprio software ad un'architettura che si porta sulle spalle il peso dei suoi vent'anni, senza tener conto delle diverse innovazioni che ogni casa ha apportato ai propri instruction set. La direzione opposta è quella di compilare per uno specifico modello di processore, traendo vantaggio dalla possibilità di appoggiarsi su specifiche sicure e avanzate, ma tagliando inesorabilmente fuori la vasta utenza che non fa uso dell'architettura di volta in volta prevista, o dissanguandosi in un versioning capillarmente differenziato, poco manutenibile e molto pesante. Tutte queste problematiche introducono al fatto che non esiste una scelta assoluta, in merito di compilatori, ma questa va progettata ad-hoc, valutando ogni volta (quantomeno) gli strumenti disponibili, le architetture di destinazione, le prestazioni desiderate e l'utenza di riferimento. Fra il ventaglio di possibilità disponibili, sarebbe certo un errore non prendere in considerazione il compilatore di uno dei leader indiscussi nel campo dei processori: l'Intel C++ Compiler (ICC).

INTEL COMPILER

ICC presenta un numero di caratteristiche che lo rendono un prodotto interessante:

- È disponibile sia per Linux(gratuito per progetti non commerciali), sia per Windows.
- Si integra automaticamente con gli IDE Eclipse (Linux) e Visual Studio (Windows).
- Ha ottime prestazioni su processori Intel, e permette di trarre vantaggio dalle diverse caratteristiche di ogni singolo modello (sfruttando i relativi instruction-set, dall'MMX fino alle recenti SSE3).
- Offre un'elegante via d'uscita al succitato problema scalabilità/specificità, grazie al meccanismo del dispatching.
- Permette di compilare per architetture Intel 64-Bite Itanium-based (1 e 2).
- Si adatta in modo naturale ad essere abbinato ad altri strumenti avanzatinelle aree che più necessitano di prestazioni elevate:

tre esempi su tutti sono rappresentati dalle OpenMPper la programmazione parallela, OpenCV, per il riconoscimento visuale e le IPP (Intel Performance Primitives) una libreria che permette di aumentare le prestazioni in una vasta gamma di applicazioni.

La combinazione con uno o più di questi strumenti è sicuramente il modo migliore di spingere a pieno regime la potenza del compilatore. Proprio per questo motivo, introdurrò in queste pagine l'uso del compilatore e, in un successivo appuntamento, mostrerò l'integrazione con le IPP. Impareremo così i principi fondamentali di questi due potenti strumenti: dalla loro installazione e integrazione in IDE, ad una panoramica sommaria delle loro funzioni.

INSTALLAZIONE DEL COMPILATORE

Come accennato precedentemente, il compilatore C++ dell'Intel è disponibile sia per Linux (dov'è gratuito per programmi non-commerciali) sia per Windows, al costo di circa 400$ o in versione trial per 30 giorni. In questa sede ci occuperemo dell'installazione del compilatore in ambiente Windows, e della sua integrazione con Visual Studio 6. Con pochi cambiamenti, lo stesso discorso può essere facilmente trasposto per i possessori di un sistema Linux (dove possibile vedremo anche le varianti Linux di quanto esposto). Il primo passo consiste nello scaricare la versione dimostrativa del compilatore (127 Mb), all'indirizzo *http://www.intel.com/software/products /compilers/downloads/cwin_eval.htm* e installare il prodotto, previa registrazione alla pagina *https://registrationcenter.intel.com/EvalCenter /EvalForm.aspx?ProductID=287*. Quest'ultima è necessaria, per ottenere il numero di serie e della chiave di licenza, che vengono inviati via e-mail (il numero di serie è nel testo e la chiave è allegata), e sono richiesti per sbloccare l'installazione del kit "Intel® C++ Compiler 8.1 for Windows*". Una volta portata a termine quest'ultima, ritroverete nel menu d'avvio

```
Build Environment for IA-32 applications
C:\>icl hello_world.cpp
Intel(R) C++ Compiler for 32-bit applications, Version 8.1    Build 20041019Z
ckage ID: W_CC_PU_8.1.020
Copyright (C) 1985-2004 Intel Corporation.  All rights reserved.
30 DAY EVALUATION LICENSE

icl: NOTE: The evaluation period for this product ends on any date UTC.
hello_world.cpp
Microsoft (R) Incremental Linker Version 6.00.8168
Copyright (C) Microsoft Corp 1992-1998. All rights reserved.

-out:hello_world.exe
hello_world.obj

C:\>_
```

Fig. 1: *Output della compilazione del programma Hello_World.cpp*

la voce *"Intel(R) Software Development Tools"*, al cui interno è presente tutto il necessario per iniziare. Per fare una prova, possiamo usare il compilatore via riga di comando (il modo più semplice e pratico), selezionando "Build Environment for IA-32 applications". Questo ci permette di accedere al prompt dei comandi, con tutte le variabili d'ambiente già inizializzate in maniera automatica e corretta. Tutto quel che ci resta da fare è semplicemente creare il nostro file e compilare. Compilare un ipotetico file "Hello_World.cpp" col minimo delle opzioni, ad esempio, corrisponde semplicemente a passare il comando:

```
> icl Hello_World.cpp
```

a seguito del quale riceveremo la pronta risposta del compilatore, ovverosia qualcosa di simile a quanto riportato in **Figura 1**. Il nostro hello_world.exe è il primo programma generato dal compilatore!

USO DEL COMPILATORE VIA CLI

Possiamo usare il prompt dei comandi per cominciare ad introdurre qualche opzione del compilatore, e mostrarne l'efficacia.
Immaginiamo di avere un programma chiamato test.cpp, come quello che segue:

```
#include <stdio.h>
#include <time.h>
#define MAX_I 2500
#define MAX_J 2500
float   n[MAX_I];
float average(float i1, float i2)
{
   return float(i1+i2)/2.0;
}
main()
{
        //inizio timer
        clock_t lStart, lEnd;
        lStart = clock();
        //computazione di prova
        for (long i=1; i<MAX_I; i++) {
                float f=0;
                for (long j=1; j<MAX_J; j++) {
                        f += average(f,j);
                        if (int(f)%2)
                           f += average(f,i);
                }
                n[i]=f;
        }
        //fine timer
```

L'IA-32
(altrimenti noto come i386), nasce nel 1985, ed è usato ancora oggi come ISA di base, per la garantire il livello minimo di compatibilità fra CPU appartenenti alla famiglia x86. Si basa su otto registri "general purpose" a 32 bit.

MMX
(possibile acronimo di MultiMedia eXtension) è il primo Instruction-Set Intel (Pentium II) di tipo SIMD, dotato di un vettore di otto registri a 64 bit.

```
        lEnd = clock();
        printf("%f secondi", (double)(lEnd-lStart)
                                    /CLOCKS_PER_SEC);
}
```

Si tratta di un piccolo test che calcola una funzione matematica priva di utilità pratica, ma che ha il grande pregio di operare una buona varietà di operazioni su numeri in virgola mobile, di "float compare and branch", di conversioni da float a int e viceversa, e di chiamate a funzione (average). La funzione potrebbe apparire singolare, ma è stata progettata ad-hoc per trarre vantaggio dalle caratteristiche più avanzate delle architetture Intel più recenti, e far risaltare particolarmente l'effetto delle ottimizzazioni che seguiranno.
Proviamo, per cominciare, a compilarla con il minimo delle opzioni, passando il comando:

```
> icl test.cpp
```

La schermata risultante (simile a quella in **Figura 1**), ci dirà che la compilazione è pronta, e che ha generato i file posti di seguito a "–out:": test.obj, e il relativo test.exe, quest'ultimo ottenuto tramite il linker Microsoft.
Possiamo provare a fare girare il file test.exe e vedere quanto impiega l'esecuzione: su un Pentium 4 a 2400 Mhz con 256 Mb di RAM il responso è circa 10.3 secondi. Un tempo che vogliamo migliorare avvalendoci di ottimizzazioni mirate.
In primo luogo, si può osservare che il codice spende la maggior parte del tempo nel ciclo più interno, nel quale esegue un numero significativo di chiamate alla funzione average. In questi casi ci si può avvalere dell'Inter-Procedural Optimization (IPO), che cerca di espandere le funzioni e ottimizzare le relative chiamate. Con il seguente comando:

```
> icl –Qip test.cpp–o:testIpo.exe
```

creiamo un file con ottimizzazione interprocedurale (opzione -Qip), di nome testIpo.exe (opzione -o:). Sempre sullo stesso Pentium 4, il risultato impiega circa 2.8 secondi: è quasi quattro volte più veloce della versione non-ottimizzata.

Windows	Linux	Effetto
-Qip	*-ip*	Applica le ottimizzazioni interprocedurali.
-QxP	*-xP*	Ottimizza per Prescott (SSE3).
-QxB	*-xB*	Ottimizza per Pentium M e compatibili.
-QxN	*-xN*	Ottimizza per Pentium IV e compatibili.
-QxK	*-xK*	Ottimizza per Pentium III e compatibili
-Qax[P|B|N|K]	*-ax[P|B|N|K]*	Come Qx*, con CPU dispatching.

Tabella 1. Un elenco delle opzioni di compilazione

DISPATCHING

Tramite l'IPO, siamo arrivati a ridurre il tempo di esecuzione di circa un quarto. Dal momento che più volte abbiamo detto di effettuare i nostri test su un Pentium 4, e che la funzione è stata creata con quest'architettura in mente, potremmo pensare di ottimizzare il risultato specificamente per questo processore. Dalla **Tabella 1**, che elenca anche un buon numero di opzioni di compilazione per altri modelli specifici, ricaviamo che lo switch da usare è –QxN. A riga di comando, quindi, passiamo:

```
> icl –Qip –QxN test.cpp–o:testIpoP4.exe
```

Su un Pentium 4, otterremo un risultato entusiasmante: il programma impiega circa 0.1 secondi, ovverosia 100 volte meno della nostra prima compilazione!
Si tratta di una gioia che potremo dividere con tutti coloro che possiedono lo stesso modello per il quale abbiamo compilato. E, purtroppo, solo con loro! Se facciamo girare il programma su un processore non compatibile con l'architettura di destinazione non otterremo altro che un laconico ed inappellabile messaggio: "Fatal Error: This program was not built to run on the processor on your system". Addio compatibilità!
Si tratta, come additavo nel paragrafo introduttivo, di un problema tipico dei compilatori: o si usano strutture avanzate e specifiche (e veloci), oppure ci si accontenta di prestazioni più modeste ma più compatibili. Nello stesso paragrafo, però, accennavo anche di un'interessante soluzione che ICC mette a disposizione dello sviluppatore: il CPU dispatching.
Quando si sceglie di usare un'opzione che impiega questo sistema, il compilatore genera due diverse versioni di ogni funzione ottimizzabile, e memorizza sia quella basata sull'architettura specifica, sia quella più lenta, ma compatibile con lo standard i386.
Quando l'eseguibile risultante viene avviato, il programma verifica automaticamente se il sistema supporta il modello per il quale si è ottimizzato, e si comporta di conseguenza. In questo modo si può ottenere, al prezzo di una minima riduzione di performance per il runtime checking, un file singolo che gira veloce sulle macchine previste, e che gira ugualmente (con minor prestazioni), su un sistema differente.
Per usare il dispatching basta trasformare il prefisso "-Qx" in "-Qxa". Quindi con la seguente riga:

```
> icl –Qip –QxaN test.cpp –o:testIpoP4_D.exe
```

GLOSSARIO

SSE
(Streaming SIMD Extension), è un'estensione avanzata all'instruction set MMX, introdotta con il Pentium III. Include otto nuovi registri a 128 bit (da xmm0 a xmm7), utilizzabili per memorizzare, al massimo, quattro numeri in virgola mobile.

SSE2
È un'estensione a SSE, introdotta con il Pentium IV con l'intenzione di soppiantare completamente MMX, che permette (fra le altre cose) l'uso di numeri in virgola mobile a doppia precisione sui registri e operazioni su interi a 64 bit.

compileremo il file con le stesse opzioni di prima, con l'aggiunta del dispatching. Il risultato gira sempre in 0.1 secondi con un Pentium 4 (in questo caso la velocità non ha risentito dei controlli runtime) e, in più, il nostro programma è sempre compatibile con architetture i386, per le quali vale comunque l'ottimizzazione interprocedurale "-Qip".
Il meccanismo di CPU dispatching avviene in automatico, con la possibilità di definire fino a tre possibili percorsi. La vera potenza di questo sistema, però, si apprezza nella versione manuale: potendo scrivere fino a sette versioni della stessa funzione, ognuna per un diverso tipo d'architettura.
Vedere in dettaglio questa funzionalità, purtroppo, richiederebbe ben più spazio di quanto racchiuso fra queste poche pagine. Per analizzare degli esempi di dispatching manuale dei processori, così come per studiare le decine d'altre opzioni che il compilatore permette di usare (e che vale la pena di conoscere a fondo), è possibile consultare il manuale in linea, e i relativi tutorial.

UNO SGUARDO VERSO IL BASSO

Immaginiamo di avere un codice come quello che segue:

```
#include <stdio.h>
#define MAX_I 10000
main()
{
        long n[MAX_I];
        printf("Inizio computazione!");
        for (int i=0; i<MAX_I; i++)
            n[i] = i;
        printf("Fine computazione!");
}
```

È abbastanza evidente che si tratta di un programma che inizializza un vettore di MAX_I elementi con un indice crescente.
Un'applicazione tanto semplice ci permette di dare un'occhiata al codice assembler generato dal compilatore, senza troppo timore.
Per raggiungere questo scopo, possiamo passare la direttiva:

```
> icl –Fa –O1 test.cpp
```

In questo caso, lo switch -O1 indica di contenere le ottimizzazioni al minimo (rendendoci, così, i passaggi più chiari), mentre –Fa richiede in output il file test.asm contenente il codice assembler generato.
Aprendolo col nostro editor preferito otterremo il listato, di cui questo è l'estratto saliente, relativo al ciclo:

```
$B1$2:                          ; Preds $B1$8
  xor       eax, eax            //i=0;          ;10.11
                             ; LOE eax ebx esi edi
$B1$3:                          ; Preds $B1$3 $B1$2
  mov       DWORD PTR [esp+eax*4], eax
                                       //n[i] = i;    ;11.3
  inc       eax                 //i++;          ;10.25
  cmp       eax, 10000          //if (i<10000) ;10.2
  jl        $B1$3     ; Prob 99%  //goto $B1$3;  ;10.2
```

Credo che il codice sia facilmente comprensibile: per maggiore chiarezza, ad ogni modo, ho anche aggiunto dei commenti "in stile C++" a fianco di ogni istruzione. In questo piccolo estratto è possibile notare la suddivisione del codice in blocchi (BXY), la corrispondenza a destra con le righe del file sorgente (ad esempio, 10.11, indica "riga 10, colonna 11"), la lista dei registri Live On Exit (LOE), la lista dei predecessori (Preds), e l'indice di probabilità associato automaticamente a ciascun'istruzione di salto (Prob).
Guardare il codice assembler generato dal compilatore è il modo migliore per capire cosa succede in seguito ad un'ottimizzazione. Proviamo, ad esempio, a ricompilare test.cpp, stavolta senza la limitazione imposta da -O1 (ovvero scrivendo solo "icl –Fa", dal momento che –O2 è lo standard):

```
$B1$2:                          ; Preds $B1$8
        xor      esi, esi       ;10.11
        mov      ebx, 1         ;10.11
        mov      ecx, 2         ;10.11
        mov      edx, 3         ;10.11
        mov      eax, 4         ;10.11
      ; LOE eax edx ecx ebx esi edi
$B1$3:                  ; Preds $B1$3 $B1$2
        mov   DWORD PTR [esp+esi*4], esi    ;11.3
        mov   DWORD PTR [esp+esi*4+4], ebx  ;11.3
        mov   DWORD PTR [esp+esi*4+8], ecx  ;11.3
        add   ebx, 5            ;10.25
        add   ecx, 5            ;10.25
        mov   DWORD PTR [esp+esi*4+12], edx ;11.3
        mov   DWORD PTR [esp+esi*4+16], eax ;11.3
        add      edx, 5     ;10.25
        add      eax, 5     ;10.25
        add      esi, 5     ;10.25
        cmp      esi, 10000  ;10.2
        jl       $B1$3          ; Prob 99%    ;10.2
```

Se non siamo abituati a certe ottimizzazioni, forse ci sembrerà strano che il codice si sia

SSE3
introdotta con la revisione Prescott del Pentium IV, è un'estensione di SSE2. Fra le altre innovazioni, permette di lavorare orizzontalmente sui registri xmm.

OPENMP
È uno standard di API per la programmazione in Fortran, C e C++ in architetture parallele. Usato in una vasta gamma di applicazioni e di sistemi operativi (fra i quali Linux, Unix e Windows), permette di scrivere applicazioni per un raggio di macchine che va dal PC al supercomputer, ai cluster per il calcolo parallelo.

OPENCV

È una libreria open-source molto potente e versatile mirata principalmente al riconoscimento visuale: è usata in una vasta gamma di applicazioni che va dal riconoscimento facciale e delle impronte digitali alla robotica. È basata sulle Intel Performance Primitives.

IPP

"Intel Performance Primitives" è una libreria Intel che permette al programmatore di disporre di una serie di funzioni atomiche e molto veloci che consentono di migliorare le prestazioni in una gamma di applicazioni che comprende la manipolazione dei segnali, delle immagini e delle matrici.

IDE

ICC si integra con Eclipse (Linux) e Visual Studio (Windows). È anche compatibile con Visual Studio.NET, sebbene manchi ancora il supporto alle managed estensions, all'event handling, all'attribute code, e, in generale, l'integrazione con tutti quei progetti specificamente marcati come .NET.

allungato, anziché accorciarsi! A dimostrazione che l'identità "lungo == lento" è quantomeno fuorviante, il compilatore ha "srotolato" il ciclo. Per farlo, si è appoggiato su più registri (eax, ebx, ecx, edx ed esi), incrementandoli di cinque unità per volta. Questo permette di ridurre di un quinto il numero dei salti, velocizzando così l'esecuzione. Il "loop unrolling" fa parte del gruppo di ottimizzazioni attivabili con –O2, e può essere personalizzato con la direttiva "-Qunroll[n]", dove n è il livello desiderato.

AUTO-VECTORIZATION

Nelle analisi precedenti, abbiamo visto solo comparire istruzioni perfettamente aderenti all'IA-32. Tutte le ottimizzazioni fatte, quindi, gireranno sulla maggior parte delle CPU, ma sappiamo che gli incrementi di prestazioni maggiori arrivano quando si sfruttano istruction-set più avanzati.
Il concetto che sta dietro queste architetture è quasi sempre lo stesso, ed è sintetizzabile con un acronimo: SIMD (Single Instruction, Multiple Data), ovvero singole istruzioni capaci di sfruttare il parallelismo del processore (quando c'è) per eseguire con un solo ciclo un'operazione su più dati. Il compilatore Intel in grado di avvalersi degli istruction-set MMX, SSE, SSE2 e SSE3 in modo proficuo (a differenza di GCC, che ancora manca quest'obiettivo), dal momento che permette una gestione della memoria già allineata in maniera corretta e offre supporto per la "auto-vectorization", ovvero il riconoscimento automatico di quei cicli in cui è possibile applicare con successo istruzioni parallele.
Possiamo provare a trarne un saggio, compilando il nostro test.cpp del paragrafo precedente, con le seguenti opzioni:

```
> icl –O1 –QxP test.cpp
```

Come sappiamo, -O1 impedisce alcune ottimizzazioni che complicherebbero l'analisi (loop unrolling), e –QxP indica al compilatore di ottimizzare per Prescott, rendendo lecita l'auto-vectorization. La risposta del compilatore sarà simile alle solite, ma conterrà una riga in più:

```
"test.cpp(10) : (col. 2) remark. LOOP WAS VECTORIZED."
```

È il segnale che il compilatore si è accorto che il nostro ciclo d'assegnamento è vettorizzabile, ed è riuscito a trasformarlo.

```
...
_RDATA  SEGMENT DWORD PUBLIC FLAT 'DATA'
_2il0floatpacket$1  DD 000000004H,000000004H,
                        000000004H,000000004H
_2il0floatpacket$3  DD 000000000H,000000001H,
                        000000002H,000000003H
_RDATA  ENDS
...
$B1$2:                          ; Preds $B1$8
   movdqa    xmm0, XMMWORD PTR
                    _2il0floatpacket$1          ;10.25
   movdqa    xmm1, XMMWORD PTR
                    _2il0floatpacket$3          ;10.25
      xor       eax, eax    ;10.2
      ; LOE eax ebx esi edi xmm0 xmm1
$B1$3:          ; Preds $B1$3 $B1$2
  movdqa  XMMWORD PTR [esp+eax*4], xmm1 ;11.3
      paddd     xmm1, xmm0              ;10.25
      add       eax, 4              ;10.2
      cmp       eax, 10000          ;10.2
      jb        $B1$3           ; Prob 99%          ;10.2
     ; LOE eax ebx esi edi xmm0 xmm1
```

E questa è la trasposizione. Ho riportato solo gli stralci significativi per la comprensione del ciclo. Innanzitutto si può notare la dichiarazione di due "pacchetti" dati, impostati a [4,4,4,4] e [0,1,2,3]. In $B1$2, al solito, è presente l'inizializzazione delle variabili: vengono caricati nei registri xmm0 e xmm1 i due pacchetti, ed eax viene posto a zero. Le cinque righe di $B1$3, invece, contengono il cuore del ciclo e due istruzioni non IA-32: Paddde movdqa, infatti, sono istruzioni SSE2, e sfruttano l'architettura parallela del processore per eseguire rispettivamente somme e spostamenti di quattro elementi di 32 bit, con una sola istruzione.
Seguiamo la prima iterazione del ciclo per comprenderne bene il funzionamento.

- **Riga 1:** Nei primi quattro elementi di n a partire da eax, vengono memorizzati i quattro di *xmm1*. (*eax = 0*, quindi: *n[0]=0, n[1]=1, n[2]=2, n[3]=3*).
- **Riga 2:** Ogni elemento di xmm1 viene incrementato di 4, tramite somma con *xmm0*. (infatti, *xmm1 += xmm0 = [0,1,2,3] + [4,4,4,4] =[4,5,6,7]*)
- **Riga 3:** *eax* viene incrementato di quattro unità. La prossima iterazione varrà per i quattro elementi di *n* successivi *(eax = 4)*.
- **Riga 4 e 5:** Se *eax < 10000* ripete il ciclo.

Tutto questo permette di velocizzare il ciclo (almeno) di un fattore 4, sfruttando l'architettura parallela dei processori Pentium successivi al IV.

Potete provare a compilare il programma senza la restrizione –O1, e vedere l'integrazione automatica fra "loop-unrolling" e "auto-vectorization".

ICC IN VISUAL STUDIO 6

Attraverso la shell si può fare di tutto, ma aiutarsi con un buon IDE è sicuramente più pratico. L'integrazione di ICC in Visual Studio è resa molto semplice dallo strumento *"Intel Compiler Selection"* che viene automaticamente aggiunto all'IDE di Visual Studio, nel menù Tools, a seguito dell'installazione. Questa voce permette di accedere alla finestra di selezione (visibile in **Figura 2**), all'interno della quale è possibile scegliere se abilitare o meno il compilatore, e per quale versione (normalmente, si vorrà usare la 8.0), sia il normale 32

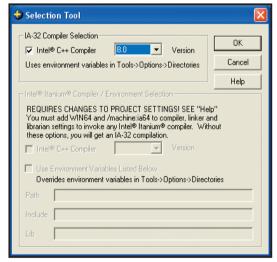



Fig. 2: *La finestra che ci consente di abilitare o meno il compilatore*

bit (nella parte alta della finestra), o il 64 bit per Itanium (in quella bassa).
Alla successiva compilazione, i file saranno processati con l'ICC, invece che col compilatore Microsoft: il linker, infatti, viene sostituito dal file "xilink6.exe", che si prende cura di effettuare questo passaggio. Come sempre, le opzioni dirette al compilatore possono essere raggiunte attraverso il menù Projects /Settings, dalla cui finestra possono essere scelti gli switch da passare alla riga di comando. Una delle caratteristiche più interessanti di questo sistema, è che si può agevolmente trarre vantaggio dal meglio di entrambi i compilatori: è infatti possibile scegliere di compilare solo alcuni file del progetto mediante ICC ed altri con il compilatore Microsoft. Per raggiungere quest'obiettivo è sufficiente:

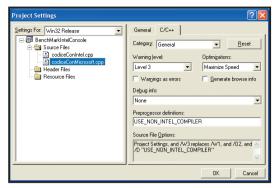



Fig. 3: *Tramite questa finestra si può scegliere se utilizzare o meno il compilatore di intel*

1. Selezionare il compilatore cui si desidera destinare la maggior parte dei file, con l'apposito Tool.

2. Andare su *Project/Settings…*

3. Selezionare il file che si vuole compilare mediante l'altro compilatore, nella lista a sinistra (come mostrato in **Figura 3**).

4. Indicare al preprocessore *"USE_NON_ INTEL_COMPILER"*, nel caso si voglia usare Microsoft (è il caso in figura), oppure *"USE_ INTEL_COMPILER"* in caso contrario.

Se si esclude l'attività di debugging diretta con ICC (per la quale è possibile usare l'apposito debugger, in versione GUI o a linea di comando), queste semplici direttive permettono di usare senza problemi l'Intel Compiler in ambiente Visual Studio.

CONCLUSIONI

Lascio nelle vostre mani un confronto fra il compilatore dell'Intel, GCC e Visual C++: un paragone che mi sono guardato bene dal proporre in questa sede.
Le tabelle dei benchmark dei compilatori, infatti, per quanto possano apparire professionali, sono notoriamente poco indicative e rischiano sempre di essere poco oggettive. Molto meglio, invece, affinare in modo pratico la propria competenza, sapendo per esperienza quando usare uno strumento, invece che un altro. L'ICC, con il supporto alla programmazione parallela, l'autovectorization, il dispatching automatico e manuale, l'ottimizzazione per profili, la compatibilità con Visual Studio ed Eclipse, la licenza free in ambito non-commerciale Linux, e molte altre qualità… è un attrezzo da tenere in bella mostra nella propria cassetta degli attrezzi

Roberto Allegra

PGO

La Profile Guided Optimization (PGO) è un tipo di ottimizzazione "multi-pass" che si può richiedere a ICC mediante le direttive –Qprof_gen e –Qprof_use. Queste permettono di ottimizzare il codice dinamicamente cercando di far capire al compilatore in automatico quali ottimizzazioni sono più indicate, attraverso più esecuzioni di prova.

Per qualsiasi critica, suggerimento o richiesta l'autore può essere contattato all'indirizzo roberto.allegra@ioprogrammo.it.

Una WebMail per cellulari con JavaMail

Spedire e ricevere posta elettronica è ormai una necessità per ogni applicazione che si rispetti. Vediamo quali sono le classi messe a disposizione dal linguaggio di Sun per questo scopo

Per il prossimo anno gli analisti stimano in 60 miliardi il numero di email scambiate giornalmente. Ogni programmatore deve tenere bene in mente che è altamente probabile che nei propri programmi avrà necessità di inserire un modulo per l'invio o la ricezione delle email. Considerate il più semplice dei casi: dare un feedback all'utente per la ricezione di un ordine.
In ogni linguaggio che si rispetti esistono delle librerie che permettono allo sviluppatore di interfacciarsi con i server di posta e consentire di inserire nei propri programmi funzioni relative alla gestione dell'email.
In Java le librerie che assolvono a questo scopo sono le JavaMail. Si tratta di un framework sviluppato dalla SUN che permette la comunicazione tramite i protocolli classici: POP3, SMTP e IMAP4. JavaMail è perfettamente compatibile con l'RFC 822, ovvero con lo standard email per Internet e questo consente di sviluppare applicazioni senza preoccuparsi di come la comunicazione con i vari server avviene.

INVIO EMAIL

Il nostro primo pezzo di codice assolverà al compito di mostrare all'utente un prompt attraverso il quale fornire le informazioni relative alla spedizione dell'email: server SMTP, destinatario, mittente, oggetto dell'email, corpo dell'email

```
import java.util.*;
import javax.mail.*;
import javax.mail.internet.*;
public class InvioEmail
{
  public static void main(String args[])
  {
    try
```

```
    {
      if (args.length<5)
      {
        System.out.println("Inserire i parametri SMTP
          DESTINATARIO MITTENTE OGGETTO CORPO");
        System.exit(0);
      }
      else
      {
        String smtp=args[0];
        String destinatario=args[1];
        String mittente=args[2];
        String oggetto=args[3];
        String corpo=args[4];
      }
    }
    catch (Exception ex)
    {System.out.println(ex.toString());}

    invia(smtp, destinatario, mittente, oggetto, corpo);
  }
}
```

Abbiamo raccolto tutti i parametri necessari per l'invio dell'email. Non ci rimane che implementare il metodo *invia(...)* che sfrutterà le classi *JavaMail* contenute nei package *javax .mail* e *javax.mail.internet*.
Prima di tutto definiremo un oggetto *Properties*, all'interno del quale imposteremo l'host del server SMTP da utilizzare. Successivamente inizializzeremo un oggetto *Session*, passandogli come parametro le variabili che sono state messe all'interno dell'oggetto *Properties*. *Session* è una classe importante all'interno di JavaMail, perché rappresenta una sessione di scambio con una casella email e contiene tutte le proprietà e i metodi necessari a stabilire la comunicazione.
Proprio grazie all'oggetto *Session* possiamo creare il messaggio da inviare, settando con gli opportuni metodi set tutte le informazioni necessarie per l'invio dell'email

```
public static void invia(String stmp,String destinatario,
            String mittente, String oggetto, String corpo)
{
  try
  {
    Properties props = System.getProperties();
    props.put("mail.smtp.host", smtp);
    Session session = Session.getDefaultInstance(
                                            props, null);
    Message msg = new MimeMessage(session);
    msg.setFrom(new InternetAddress(mittente));
    msg.setRecipients(Message.RecipientType.TO,
          InternetAddress.parse(destinatario, false));
    msg.setSubject(oggetto);
    msg.setText(corpo);
    msg.setHeader("X-Mailer", "ioProgrammoMailer");
    msg.setSentDate(new Date());
    // Finiti i vari settaggi avviene la consegna del
                            messaggio email al metodo
    // statico "send" della class Transport, la quale
                            instraderà la nostra email sul
    // server che abbiamo definito
    Transport.send(msg);
    System.out.println("Messaggio inviato
                                    correttamente");
  }
  catch(Exception e)
  {
      System.out.println("Errore nell'invio dell'email");}
}
```

L'invio di un'email è un compito abbastanza semplice grazie a JavaMail. Senza queste classi avremmo dovuto provvedere a programmare da soli una soluzione per il collegamento con il server SMTP. Avremmo dovuto avere conoscenza delle l'RFC *(Requests for Comments)*, e avremmo dovuto inviare via Socket tutte le informazioni e parsare i messaggi di risposta del server. Utilizzando JavaMail, invece, dobbiamo soltanto preoccuparci di riempire i campi necessari dell'email e consegnarla all'oggetto *Transport* che svolgerà tutto il lavoro. Per provare questa piccola classe dobbiamo includere nel nostro classpath *mail .jar* e *activation.jar*, i due file relativi rispettivamente a JavaMail e a JAF *(Java Activation Framework)*.

LETTURA EMAIL

Per poter scaricare la posta da un server dovremo fornire un indirizzo, uno username e una password. Sempre grazie all'oggetto *Session*, inizializzeremo uno *Store* che rappresenterà il repository dei nostri messaggi. In questo caso ci collegheremo ad uno Store POP3, ma JavaMail permette di collegarsi anche a server IMAP4. Il codice seguente illustra la procedura per effettuare la connessione e stampare a schermo i messaggi ricevuti

```
public static void ricevi(String pop3, String
                        username, String password) {
  Store store=null;
  Folder folder=null;
  try {
    Properties props = System.getProperties();
    Session session = Session.getDefaultInstance(
                                            props, null);

    store = session.getStore("pop3");
    store.connect(pop3, username, password);
    // Una volta collegati carichiamo la directory INBOX
    // della nostra posta elettronica. Inoltre settiamo
                            la cartella come READ_ONLY
    // per indicare che stiamo per leggere soltanto l'email
    folder = store.getDefaultFolder();
    folder = folder.getFolder("INBOX");
    folder.open(Folder.READ_ONLY);
    Message[] messaggi = folder.getMessages();
    System.out.println("Ci sono "+messaggi.length+"
        messaggi di posta nella tua casella email");}
  catch (Exception e) {
    System.out.println(e.toString());}
  finally {
    // Cerchiamo di chiudere il Folder e lo Store,
    // anche perchè ci sono alcuni server POP3 che
    // lasciano bloccata l'email se non effettuiamo
    // questa operazione correttamente.
    try {
      if (folder!=null) folder.close(false);
      if (store!=null) store.close(); }
    catch (Exception ee) {
      System.out.println(ee.toString());}
  }
}
```

Anche la connessione con il server POP3, come appena mostrato non comporta nessun problema. All'interno di JavaMail è presente il package *com.sun.mail.imap*, che permette di interagire anche con i server IMAP4. Per quest'ultimo protocollo bisogna ricordarsi della possibilità di creare cartelle diverse da *INBOX*. Infatti con un account IMAP4 possiamo avere a disposizione diverse cartelle, che possiamo richiamare come già abbiamo fatto per la casella POP3

```
store = session.getStore("imap");
store.connect(imap, username, password);
folder = store.getDefaultFolder();
folder = folder.getFolder("EmailVecchie");
folder.open(Folder.READ_ONLY);
```

SMTP (SIMPLE MAIL TRANSFER PROTOCOL)
È il protocollo standard per l'invio di email su Internet. Attraverso un dialogo che avviene tra client e server, si riesce ad inviare un messaggio di posta elettronica, specificando tutti i vari attributi che sono necessari per l'invio.
http://www.ietf.org/rfc/rfc0821.txt

POP3 (POSTAL OFFICE PROTOCOL 3)
È un protocollo per la ricezione della posta su internet. In seguito ad una necessaria autenticazione del client è possibile consultare i messaggi di posta arrivati sul server, scaricarli e cancellarli
http://www.ietf.org/rfc/rfc1939.txt

IMAP4 (INTERNET MESSAGE ACCESS PROTOCOL 4)
È un ulteriore protocollo per la gestione della posta in arrivo. Oltre alle funzioni del POP3, permette di organizzare delle vere e proprie cartelle all'interno del server di posta
http://www.ietf.org/rfc/rfc1730.txt

Sono disponibili diverse implementazioni anche per interfacciare le JavaMail API con un server NNTP *(Network News Transfer Protocol)*, ovvero un server di news. Queste implementazioni permettono di richiamare all'interno dell'oggetto *Session* il server NNTP e di gestire la connessione verso i newsgroup in maniera completamente uguale alla consultazione della casella di posta.
http://www.ietf.org/rfc/rfc977.txt
http://bluezoo.org/knife/
http://www.vroyer.org/lgpl/

La specifica di JavaMail non tratta in alcuna maniera la protezione dei dati trasmessi. Per fare ciò possiamo utilizzare JSSE (Java Secure Socket Extension), un insieme di package che permette di utilizzare connessioni sicure SSL e TLS.
http://www.javaworld.com/javatips/jw-javatip115.html
http://java.sun.com/products/jsse/

WEBMAIL PER CELLULARI

I cellulari di ultima generazione hanno la possibilità di navigare su internet grazie alla presenza di una versione ridotta dei browser al loro interno. Potremmo creare una webmail per consentire di leggere la posta attraverso il browser del cellulare. L'applicazione è abbastanza semplice. È suddivisibile in due diverse parti funzionali, invio e lettura email.
Purtroppo per il momento la nostra webmail leggerà soltanto le email testuali, altrimenti dovremmo parsare l'html. Nonostante questo, JavaMail permette di gestire diversi formati di allegati. Infatti un oggetto email può contenere diversi tipi di file e quindi avere diversi tipi di *content-type*. Attraverso l'interfaccia *Part*, definita in JavaMail, possiamo controllare di quanti parti è composta la nostra email e soprattutto vedere quali sono i content-type relativi ad ogni parte. In questa applicazione che andremo a sviluppare, prenderemo in considerazione soltanto il content-type *"text/plain"*, ovvero testo semplice.

MENU INIZIALE

Le pagine che vogliamo visualizzare nel browser WAP devono essere scritte in WML *(Wireless Markup Language)*, linguaggio standard per il markup di pagine che devono essere visualizzate su dispositivi come cellulari.
Il WML è stato pensato proprio per device con capacità computazionali limitate tipico delle applicazioni che girano all'interno di un cellulare. Tutto il lavoro di generazione dinamica delle pagine avverrà lato server, tramite una servlet che, in base al metodo richiamato, genererà pagine WML comprensibili per il nostro cellulare.
Come punto di partenza per la nostra applicazione avremo un menu nel quale faremo scegliere all'utente cosa fare

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML
                                        1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<wml><card>
<p align="center">
WebMail<br/>
<img src="Java.wbmp" alt="WebMail"/><br/>
<a href="http://mioserver/servetInvia" title"Invia
                         Email"> Invia Email </a><br/>
<a href="http://mioserver/servetRicevi" title"Ricevi
                              Email"> Ricevi Email </a>
</p>
</card>
</wml>
```

Con questa pagina WML diamo all'utente la possibilità di scegliere tra l'invio e la ricezione dell'email. Queste funzionalità verranno realizzate con 2 semplici servlet, dove dovremo soltanto formattare in WML l'output del codice visto in precedenza.

SERVLET PER L'INVIO

Nella servlet che utilizzeremo per l'invio dell'email sarà necessario suddividere due diversi momenti: la generazione statica del form WML per inserire i dati che ci servono per la spedizione e la generazione dinamica del risultato. Il metodo *doGet()* conterrà il codice WML della pagina che ci servirà per raccogliere l'input

```
public void doGet (HttpServletRequest request,
                  HttpServletResponse response) throws
                        ServletException, IOException
{
   PrintWriter out;
   response.setContentType("text/vnd.wap.wml");
   out = response.getWriter();
   out.println("<wml><card><p align='center'>
   SMTP: <input type='text' format='*M'
              emptyok='false' name='SMTP' size='10'
                              maxlength='40'/><br/>
   Da: <input type='text' format='*M' emptyok=
                    'false' name='From' size='10'
                              maxlength='40'/><br/>
   A: <input type='text' format='*M' emptyok='false'
          name='To' size='10' maxlength='40'/><br/>
   Oggetto: <input type='text' format='*M' emptyok=
          'false' name='Subject' size='10' maxlength=
                                        '100'/> <br/>
   Messaggio: <input type='text' format='*M'
              emptyok='false' name='Body' size='10'
                              maxlength='40'/> <br/>
  <anchor title='Invia'>Invia<go href='servletInvia'>
      <postfield name='SMTP' value='$(SMTP)' />
      <postfield name='From' value='$(From)' />
      <postfield name='To' value='$(To)' />
      <postfield name='Subject' value='$(Subject)' />
      <postfield name='Body' value='$(Body)' />
    </go></anchor>
    </p></card></wml>");
out.close();
}
```

L'utente visualizza un form WML da riempire per l'invio dell'email. Cliccando su "*Invia*", la nostra servlet vierrà richiamata e riceverà tut-

ti i valori necessari per l'invio. Nel metodo doPost della servlet Invia quindi dovremo semplicemente riportare il codice che abbiamo già utilizzato per l'invio dell'email. In aggiunta dovremo caricare una pagina per permettere all'utente di ritornare al menu principale con un semplice link.

SERVLET PER LA RICEZIONE

Anche in questo caso avremo bisogno di una prima schermata dove prendere le informazioni relative a server POP3, username e password. Quindi allo stesso modo dell'invio, genereremo un form WML nel metodo *doGet* per acquisisre questi dati. Una volta inseriti e inviati alla servlet via POST mostreremo all'utente semplicemente le email presenti nella sua casella di posta.

```
Store store=null;
Folder folder=null;
Flags flag;
Properties props = System.getProperties();
Session session = Session.getDefaultInstance(
                                        props, null);
store = session.getStore("pop3");
store.connect(pop3, username, password);
folder = store.getDefaultFolder();
folder = folder.getFolder("INBOX");
folder.open(Folder.READ_ONLY);
Message[] messaggi = folder.getMessages();
int nuove=0;
//Leggo quante nuove email ci sono nella INBOX
for (int i=0;i<messaggi.length;i++)
{
   flags = messaggi[i].getFlags();
   boolean letto=flags.contains(Flags.Flag.SEEN);
   if (!letto)
      nuove++;
}
out.println("
<wml><card>
<p align='center'>
Lettura Email<br/>
Ci sono "+nuove+" nuove email su
                        "+messaggi.length+" email
<a href='http://mioserver/servetLeggi' title
                  'LeggiEmail'> Leggi Email </a><br/>
<do type="Indietro" title="indietro">
 <prev/> </do></p></card></wml>"
);
```

Ora rimane da realizzare la servlet che ci permette di vedere quali email ci sono nella nostra casella elettronica e di leggerle. Dovremo visualizzare il mittente e l'oggetto per ogni email, poi l'utente, tramite click nel browser WAP, deciderà quale email visualizzare.
Quindi ecco come sarà il codice

```
String mittente="";
String oggetto="";
Message[] messaggi = folder.getMessages();
for (int i=0;i<messaggi.length;i++)
{
   mittente=((InternetAddress)messaggi[i].getFrom()
                                )[0].getAddress();
   oggetto= messaggi[i].getSubject();
   out.println("<a href='http://mioserver/servetLeggi?
                     num="+i+"'> "+ mittente +" -
                        "+ oggetto+"</a><br/>");
}
```

Per ogni email che andremo a visualizzare dovremo prendere soltanto il testo dell'email, non visualizzando altre parti come immagini e quant'altro. Succede spesso che per colpa di email non formattate secondo lo standard RFC 822, JavaMail fallisce il parsing del messaggio. In quel caso possiamo soltanto prevedere una serie di *try-catch* per evitare che salti l'intero programma.

CONCLUSIONI

Grazie al package JavaMail abbiamo visto come sia semplice ed immediato inviare e ricevere email nel linguaggio Java. Infatti questo package permette allo sviluppatore di preoccuparsi soltanto della logica di business del proprio programma, senza dover pensare ad interfacciarsi direttamente con i server di posta. In questo modo la connessione, gestione delle cartelle, degli allegati, delle email è trasparente per lo sviluppatore. JavaMail, che attualmente è arrivato alla versione 1.3.2, ha bisogno per funzionare di JAF *(Java Activation Framework)*, grazie al quale riesce a identificare e gestire i vari content-type dei messaggi email. Per utilizzare questo package dobbiamo quindi includere i due file mail.jar e *activation.jar* (rispettivamente di JavaMail e JAF) nel nostro classpath. Facendo parte della piattaforma J2EE *(Java 2 Enterprise Edition)*, possiamo utilizzare questo package semplicemente importando le classi nella nostra applicazione Enterprise.
È utile, tuttavia, anche dal punto di vista di J2SE, per la realizzazione di semplici applicazioni che si appoggiano ai server di posta.

Federico Paparoni

L'AUTORE

Federico Paparoni è laureato in Ingegneria Informatica. È specializzato nello sviluppo di applicazioni client-server per terminali mobili e collabora con delle aziende di telefonia. I suoi interessi principali riguardano le piattaforme J2ME e J2EE. È admin di *JavaStaff.com* e può essere contattato all'email federico.paparoni@javastaff.com

SUL WEB

Ecco alcuni link utili:

JavaMail
http://java.sun.com/products/javamail/

JAF
http://java.sun.com/beans/glasgow/jaf.html

Articoli tecnici della SUN su JavaMail
http://java.sun.com/products/javamail/reference/techart/index.html

Fare collezione di Oggetti in VB

Manipolare insiemi omogenei di strutture utilizzando VB.NET. Una tecnica indispensabile per aggregare i dati affinché possano essere analizzati e usati in maniera conveniente.

Conoscenze richieste
Elementi Visual Basic .NET acquisiti nel corso

Software
Sistema operativo: Windows 2000/XP. Visual Basic .NET 2003

Impegno

Tempo di realizzazione

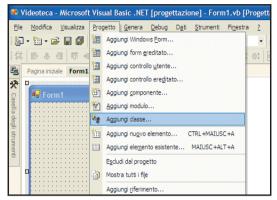
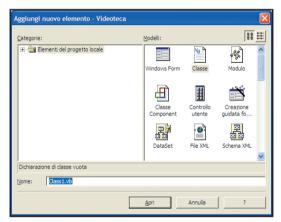
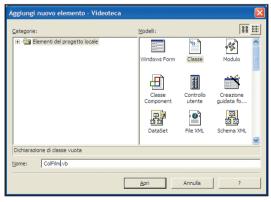
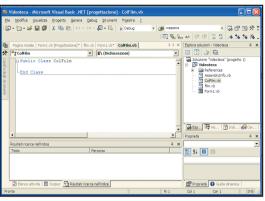
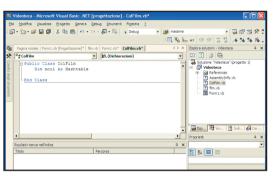
In natura i dati non si trovano quasi mai in forma singola, piuttosto vengono organizzati in insiemi omogenei. Ad esempio l'insieme degli articoli prodotti per questa edizione di ioProgrammo. L'insieme dei film di una videoteca. Ciascuno di questi dati può, come sappiamo essere rappresentato da una classe che ne definisce proprietà e metodi. È anche vero che, instanziare un oggetto come appartenente ad una classe, non vuol dire necessariamente aggregarlo in una collezione omogenea. Supponiamo ad esempio di instanziare 100 oggetti di classe *Film*, fino a che non li aggreghiamo in un qualche modo, ognuno dei 100 oggetti *Film* rappresenta un dato a sé stante. Esiste una classe in VB che definisce delle collezioni di oggetti.
Questa classe fornisce metodi e proprietà per inserire, rimuovere o manipolare oggetti appartenenti ad una collezione. Ad esempio la Collection *VideoTeca* di tutti i film instanziati, espone metodi e proprietà per inserire o rimuovere oggetti *Film* dal suo interno, e altri ancora di cui parleremo proprio in questo articolo.

IMPOSTARE UNA CLASSE COLLEZIONE PERSONALIZZATA

In VB.NET sono disponibili diverse classi collezione native. Alcune, come *Stack* e *Queue* sono classi specializzate implementate per svolgere ruoli specifici. *Stack* simula un insieme *LIFO* (*Last-in, first-out* conosciuta anche come pila), Queue un insieme di oggetti *FIFO* (*First-In First-Out* conosciuta anche come coda).
Altre classi, quali *CollectionBase* e *DictionaryBase,* sono classi astratte che presentano solo alcune funzionalità di base, mentre è lasciato allo sviluppatore, il compito di scrivere la maggior parte del codice d'implementazione. In VB.NET è possibile creare classi *Collezione* personalizzate, ereditando da una delle numerose classi collezione di .NET Framework e aggiungendo codice per l'implementazione di funzionalità necessarie ad un oggetto di tipo collezione. Nella classe *CollectionBase* sono già disponibili implementazioni per il metodo *Clear,* che permette di cancellare tutti gli oggetti di una collezione e la proprietà *Count,* che restituisce il numero di elementi presenti nella collezione. Per l'organizzazione e l'archiviazione interna degli oggetti viene mantenuta una proprietà protetta denominata *List*.
Un'altra possibilità è quella di utilizzare una classe nativa definita privata (per garantire l'incapsulamento, che abbiamo visto essere una caratteristica essenziale della programmazione ad oggetti), in una classe con il solo compito specifico di gestire una collezione di oggetti. In particolare si può utilizzare la collezione nativa *HashTable.* Si tratta di un tipo di collezione speciale che funziona in base ad un principio chiave/valore, in pratica ad ogni elemento della collezione è assegnata una chiave, che può essere utilizzata per recuperare il valore dell'oggetto corrispondente.
Così come per le proprietà di una classe, che per essere lette o modificate devono esporre le procedure *Property,* anche per le collezioni si ha la necessità di avere metodi contenitori pubblici che permettano l'accesso ad una collezione privata ed ai suoi metodi nativi.

CREARE UNA CLASSE COLLEZIONE

Per i nostri esempi ci appoggeremo all'applicazione *"Videoteca"* creata nei numeri precedenti di ioProgrammo. Chi non avesse seguito fin qui, o non dispoenesse del codice necessario a seguire l'articolo, potrà comunque trovarne una versione nel cd allegato alla rivista. In ogni caso gli esempi sono del

tutto generici per cui potrete riadattarli seconco le vostre necessità.
I passi per creare una nuova classe collezione sono i seguenti:

1 per inserire una nuova classe collezione nel progetto, dobbiamo selezionare dal menu *Progetto* la voce *aggiungi classe*. In questo modo sarà visualizzata la finestra di dialogo *Aggiungi nuovo elemento.*

2 Nella finestra di dialogo *Aggiungi* nuovo elemento, sarà selezionato il modello *Classe* (nella parte destra della maschera).

3 Nel campo *Nome* dobbiamo inserire il nome della classe. collezione. Esistono diverse convenzioni in letteratura per l'attribuzione del nome ad una collezione, personalmente preferisco far precedere il nome della classe dal prefisso *Col (ColFilm).*

4 Clicchiamo sul pulsante *Apri.* Si aprirà la finestra di progettazione del codice di VB.NET con il cursore posto all'interno della dichiarazione della classe collezione.

5 Nella finestra del codice (dopo l'istruzione di dichiarazione della classe), si deve dichiarare una variabile oggetto privata *mCol* di tipo *Hashtable.*

```
Dim mcol As Hashtable
```

Utilizza questo spazio per le tue annotazioni

IL CODICE DA SCRIVERE NEL BUTTON CANCELLA

```
Private Sub ButtonCancella_Click(
        ByVal sender As System.Object,
        ByVal e As System.EventArgs)
        Handles ButtonCancella.Click
   'Elimina l'eventuale errore che si
   'potrebbe verificare se si tenta di
   'cancellare un film non presente
   'nella collezione
   Try
   'chiamata al metodo Remove
   'passando come parametro la chiave
   'rappresentata dal titolo del film
      CollezioneDiFilm.Remove(
                  TextBoxTitolo.Text)
    Catch
    End Try
   'Visualizza la nuova lista di film
    VisualizzaLista()
   'per svuotare i TextBox
    SvuotaCampi()
   'per riportarsi nelle condizioni iniziali
    ObjFilm = Nothing
End Sub
```

Il ciclo *For Each...Next* è simile al ciclo *For...Next*, ma invece di ripetere le istruzioni il numero di volte specificato, ripete un gruppo di istruzioni per ciascun elemento di un insieme di oggetti, questo risulta particolarmente utile se non si conosce il numero di elementi di un insieme. La sintassi è la seguente:

```
For Each VariabileOggetto In
                 CollezioneDiOggetti
  'blocco di istruzioni da eseguire su
  'ogni oggetto della collezione
Next
```

A ogni ripetizione del ciclo, la variabile *VariabileOggetto* viene impostata su uno degli elementi della collezione e viene eseguito il blocco di istruzioni. Quando tutti gli elementi della collezione sono stati assegnati a *VariabileOggetto*, il ciclo *For Each* termina e il controllo passa all'istruzione successiva all'istruzione *Next*.

CREARE L'ISTANZA DELLA VARIABILE OGGETTO

Dopo aver definito la variabile collezione, come ogni variabile oggetto, si deve creare un'istanza della variabile di tipo *Hashtable*. Il posto migliore in cui creare l'istanza della variabile oggetto *mCol* è nel costruttore della classe. Il costruttore è un metodo richiamato automaticamente ogni volta che viene creata un'istanza di un oggetto, in particolare è una specifica routine *Sub* il cui nome deve obbligatoriamente essere *New.* Possiamo quindi scrivere:

```
Public Sub New()
    'crea l'oggetto di tipo Hashtable
    mcol = New Hashtable()
End Sub
```

A questo punto avremo a disposizione una classe che crea una collezione privata. Perché questa classe sia di effettivo interesse si devono aggiungere i metodi che permettano di manipolare la collezione privata. In particolare gli elementi di una collezione sono aggiunti con il metodo *Add* e rimossi con il metodo *Remove.* Uno specifico elemento della collezione può essere referenziato con il metodo *Item.*
Inoltre la proprietà Count restituisce il numero di membri della collezione. Analizziamoli in dettaglio.

METODO ADD

Poiché la variabile di tipo *Hashtable,* è stata dichiarata come variabile privata della classe, non è possibile aggiungere oggetti alla collezione da un punto qualsiasi dell'applicazione. Per questo motivo si deve definire un metodo pubblico *Add* usando una procedura *Sub* pubblica che sfrutti il metodo *Add*

DISEGNIAMO L'INTERFACCIA UTENTE DELL' APPLICAZIONE VIDEOTECA

Nel numero precedente di ioProgrammo abbiamo iniziato la costruzione di un progetto per la gestione di una videoteca. In questo numero sfrutteremo le conoscenze acquisite sulle collection per rendere più semplice la programmazione. Per quelli che non hanno seguito il numero precedente riassumiamo brevemente i passi per la creazione dell'interfaccia utente. Prevediamo un tasto "nuovo" per inserire un nuovo film, una dialog per inserire i dati, un tasto salva per salvare fisicamente i dati delll'applicazione amento dell'applicazione. Tutto si può riassumere con i passi illustrati in questa procedura guidata.

1 **Selezioniamo la finestra *FormVideoteca* realizzata nell'articolo precedente che contiene i seguenti controlli: N. 4 *TextBox* dal nome: *TextBoxTitolo, TextBoxGenere, TextBoxCosto, TextBoxGiorni.* N. 4 Label con la descrizione dei *TextBox* corrispondenti. N. 1 controllo *Button* dal nome: *ButtonMostraDati.***

2 **Selezioniamo per tre volte un controllo *Button* dalla casella degli strumenti (nella sezione *Windows Form*) e disegniamo i tre controlli sulla form. Utilizzeremo questi bottoni per gestire le varie azioni corrispondenti a "nuovo", "inserisci", "salva" e che rappresentano il cuore dell'applicazione**

3 **Selezioniamo il primo *Button* , evidenziamo la proprietà Name per cambiare il nome in: *ButtonNuovo.* Evidenziamo la proprietà *Text* e cambiamo il testo in *Nuovo.* Facciamo lo stesso con gli altri variando la proprietà *Name* in: *ButtonSalva, ButtonCancella* e le proprietà *Text* a *Salva* e *Cancella.***

4 **Selezioniamo un controllo *ListBox* dalla casella degli strumenti e disegniamolo sulla form. Lo utilizzeremo per contenere l'elenco di tutti i film presenti nella videoteca. Chiaramente più avanti illustreremo il codice necessario ad inserire o rimuovere un film da questo controllo**

5 **Selezioniamo il controllo *ListBox* e, dalla finestra delle proprietà, modifichiamo la proprietà Name in: *ListBoxFilm.* Chiaramente da questo momento in poi ogni riferimento a questo controllo potrà essere effettuato utilizzando il suo nome, così come lo abbiamo settato nelle proprietà**

nativo della collezione *Hashtable*. Il metodo *Add* nativo della collezione *Hashtable*, consente di aggiungere alla collezione un oggetto con la chiave ed il valore specificato. Per il nostro esempio si può utilizzare il titolo del film come chiave univoca per l'individuazione di un generico oggetto *Film*.

```
Public Sub Add(ByVal obj As film)
  'viene chiamato il metodo nativo Add della Hashtable
  'passandogli come parametri la chiave di accesso
  'e l'oggetto che dovrà contenere
   mcol.Add(obj.Titolo, obj)
End Sub
```

IL METODO REMOVE

Per esporre l'opportunità di rimuovere un oggetto dalla collezione, si deve definire un metodo involucro *Remove* pubblico che sfrutti il metodo *Remove* nativo.

```
Public Sub Remove(ByVal Chiave As String)
   'Utilizzata per rimuovere un elemento dalla collezione.
        'in base alla chiave dell'oggetto
        mcol.Remove(Chiave)
    End Sub
```

Il metodo *Remove* nativo elimina dalla collezione l'oggetto che è stato aggiunto con la chiave passata come argomento. Se la chiave specificata non viene trovata nella collezione, VB genera un errore di *Run-Time*. È possibile migliorare il metodo introducendo un gestore di eccezioni che eviti questo errore.

IL METODO ITEM

Il metodo involucro *Item* permette di referenziare un elemento specifico della collezione. Per definire il metodo pubblico *Item* si utilizza una *Property Get* a sola lettura che restituisce il riferimento ad uno specifico oggetto della collezione individuato dal parametro *Chiave*.

```
Default Public ReadOnly Property Item(ByVal Chiave
                                    As String) As film
     Get
          'Utilizzato per fare riferimento ad un elemento
                                         della collezione.
          'con la Chiave passata come parametro
          Return mcol.Item(chiave)
     End Get
End Property
```

La proprietà *Item* può diventare la proprietà predefinita (si può definire soltanto una proprietà all'interno di una classe come default), utilizzando nella sua dichiarazione la parola chiave *Default*. In questo modo, nel momento in cui all'interno del codice si vorrà fare riferimento ad un oggetto della collezione, si potrà omettere la chiamata a questa proprietà.
Dopo aver referenziato lo specifico oggetto della collezione, ed aver assegnato il riferimento ad una variabile oggetto, sarà possibile accedere alle proprietà ed ai metodi dell'oggetto restituito.

LA PROPRIETÀ COUNT

La proprietà *Count* dovrà restituire il numero di elementi presenti nella collezione. Per implementarla si deve definire una proprietà pubblica a sola lettura che chiami la proprietà intrinseca *Count* della collezione *Hashtable*

```
     Public ReadOnly Property Count() As Integer
         Get
             Return mcol.Count
         End Get
     End Property
```

SCORRERE GLI ELEMENTI DI UNA COLLEZIONE

I metodi e le proprietà definiti finora rappresentano il set minimale che una collezione deve implementare, per migliorare la vita al programmatore si può pensare a un metodo che consenta di navigare facilmente tra gli oggetti di una collezione: il metodo *Elements*. Il metodo *Elements* dovrà restituire un enumeratore, cioè un oggetto che scorre l'insieme associato. L'enumeratore è simile ad un puntatore che si sposta su qualsiasi elemento dell'insieme, e viene utilizzato nell'istruzione *For Each.. Next*.
L'interfaccia *ICollection* fornisce il suo enumeratore, pertanto si può scrivere:

```
Public ReadOnly Property Elements() As ICollection
        Get
            Return mcol.Values
        End Get
End Property
```

L'APPLICAZIONE VIDEOTECA

Per inserire i dati di un nuovo film, il gestore della videoteca dovrà:

- Premereil tasto *Nuovo*.
- Inserire i dati del film nei TextBox *Corrispondenti*.

Quando si aggiunge un oggetto alla collezione, esso non viene realmente aggiunto alla collezione, piuttosto viene aggiunto alla collezione un riferimento all'oggetto. Una collezione contenente un insieme di oggetti contiene, quindi, in realtà un insieme di riferimenti ad oggetti.

- Premere il tasto *Salva* per inserire il film nella lista della videoteca.

Analizziamo il codice necessario al funzionamento dell'applicazione.

```
'Definizione delle variabili globali
Dim CollezioneDiFilm As ColFilm
Dim ObjFilm As film
```

La variabile oggetto *CollezioneDiFilmrappresenta* una istanza della classe *ColFilm,* che dovrà contenere per la durata dell'applicazione l'insieme dei film della videoteca. La variabile oggetto *ObjFilm* rappresenta una istanza della classe *Film,* che dovrà contenere i dati del film corrente.

COSTRUTTORI

Un aspetto molto importante della progettazione di oggetti è il concetto di costruttore. Il costruttore è una porzione di codice di inizializzazione eseguito ogni volta che viene creata un'istanza di un oggetto. Tipiche operazioni d'inizializzazione sono: l'apertura di file, la connessione ad un database, o più semplicemente l'impostazione di valori predefiniti per le proprietà di un oggetto.

Per creare un costruttore, si deve scrivere una procedura denominata Sub New in un qualsiasi punto del codice della classe. Il codice del metodo Sub New viene eseguito prima del resto del codice della classe. Se non si definisce in modo esplicito una Sub New, VB .NET crea in modo implicito, in fase di esecuzione, un costruttore Sub New.

INIZIALIZZARE LA CLASSE COLLEZIONE

Il ciclo di vita dell'oggetto *CollezioneDiFilmcoincide* con il ciclo di vita della form per questo motivo la variabile oggetto *CollezioneDiFilmdovrà* essere creata nell'evento *Load* della form

```
Private Sub FormVideoteca_Load(ByVal sender As
          System.Object, ByVal e As System.EventArgs)
                              Handles MyBase.Load
        CollezioneDiFilm = New ColFilm
End Sub
```

IL PULSANTE NUOVO

Per inserire un nuovo film nella videoteca, la prima operazione che dovrà compiere l'utente sarà di premere il pulsante nuovo. Nell'evento *click* di *ButtonNuovo* dovremo, quindi, preoccuparci di svuotare i TextBox e di creare la variabile oggetto *ObjFilm* con la sintassi ormai nota:

```
Private Sub ButtonNuovo_Click(ByVal sender As
          System.Object, ByVal e As System.EventArgs)
                              Handles ButtonNuovo.Click
        ObjFilm = New film
        SvuotaCampi()
End Sub
```

LA PROCEDURA SVUOTACAMPI

La procedura per pulire i campi non dovrà fare altro che porre la proprietà *Text* dei quattro TextBox pari alla stringa vuota. Per questo possiamo scrivere:

```
Private Sub SvuotaCampi()
        TextBoxTitolo.Text = ""
        TextBoxGenere.Text = ""
        TextBoxCosto.Text = ""
        TextBoxGiorni.Text = ""
End Sub
```

IL PULSANTE SALVA

Quando l'utente completa l'inserimento dei dati, dovrà premere il tasto salva per inserire il film nella lista dei film della videoteca.

```
Private Sub ButtonSalva_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
                              Handles ButtonSalva.Click
        'controllo sull'esistenza dell'oggetto ObjFilm
        If ObjFilm Is Nothing Then
            MessageBox.Show("Si deve prima premere il
                                          tasto Nuovo")
            Exit Sub
        End If
        ValorizzaCampi()
        Try
            CollezioneDiFilm.Add(ObjFilm)
        Catch
        End Try
        VisualizzaLista()
        SvuotaCampi()
        ObjFilm = Nothing
End Sub
```

La prima istruzione controlla se l'oggetto *ObjFilm* è stato creato. In caso contrario avvisa l'utente che prima di inserire i dati di un nuovo film si deve premere il tasto *Nuovo,* e forza l'uscita dalla procedura. La procedura *ValorizzaCampi* dovrà assegnare i valori inseriti nei tre TextBox alle proprietà dell'oggetto *ObjFilm.*

INSERIRE IL FILM NELLA COLLEZIONE

L'istruzione: *CollezioneDiFilm.Add(ObjFilm),* inserisce il film corrente nella collezione dei film della videoteca. L'istruzione è preceduta dal gestore di eccezioni *Try..Catch..End Try,* che analizzeremo meglio in uno dei prossimi articoli. Ci basti sapere che in questa forma, cattura l'errore e passa all'istruzione successiva ad *End Try* nel caso in cui si tenta di inserire nella collezione un film con lo stesso titolo di un film già presente nella videoteca. Utilizzando l'istruzione *Try..Catch..End Try* in questa forma, si potrà usare il tasto *Salva* finanche per modificare i dati di un film selezionato tra la lista dei film della videoteca. Infine la procedura *VisualizzaLista* dovrà visualizzare nel ListBox il film appena inserito.

LA PROCEDURA VISUALIZZALISTA

```
Private Sub VisualizzaLista()
    Dim tmpFilm As New film
    ListBoxFilm.Items.Clear()
    For Each tmpFilm In CollezioneDiFilm.Elements
        ListBoxFilm.Items.Add(tmpFilm.Titolo)
    Next
End Sub
```

La variabile locale *tmpFilm*, ci permetterà di ciclare tra tutti i film della videoteca, referenziando volta per volta un oggetto film diverso. La successiva istruzione svuota il ListBox per mezzo del metodo *Clear* dell'insieme *Items*. Utilizzando l'istruzione *For Each… Next* è possibile ciclare su ogni film della collezione, ed aggiungere il titolo del film nel ListBox, tramite la proprietà *Add* dell'insieme *Items*. Questa procedura può essere resa più performante poiché, come possiamo facilmente notare, ogni volta che l'utente preme il tasto *Salva* per inserire un nuovo film, la lista viene prima svuotata e poi riempita ciclando su tutti i film della collezione. Per una videoteca poco fornita può andare bene così.

LA PROCEDURA VALORIZZACAMPI

La procedura *ValorizzaCampi* dovrà assegnare i valori inseriti nei tre TextBox alle proprietà dell'oggetto *ObjFilm*. Il codice sarà semplicemente:

```
Private Sub ValorizzaCampi()
    ObjFilm.Titolo = TextBoxTitolo.Text
    ObjFilm.Genere = TextBoxGenere.Text
    ObjFilm.Costo = CDec(TextBoxCosto.Text)
End Sub
```

MODIFICARE I DATI DI UN FILM

L'ultima funzionalità che ci resta da implementare è la possibilità di modificare i dati di un film presente nella lista (doppio click sul ListBox). Per ottenere questo scopo, l'utente dovrà selezionare il film dalla lista, modificare eventualmente i suoi dati e premere il tasto *Salva* per visualizzare i nuovi dati nella lista. Nell'evento *DoubleClick* del ListBox scriviamo:

```
Private Sub ListBoxFilm_DoubleClick(ByVal sender As
    Object, ByVal e As System.EventArgs) Handles
                                ListBoxFilm.DoubleClick
    ' Verifica che sia stato selezionato un film.
        If ListBoxFilm.SelectedItem <> "" Then
    ObjFilm = CollezioneDiFilm(
ListBoxFilm.SelectedItem)
        VisualizzaDati()
    End If
End Sub
```

Tramite la proprietà *SelectedItem* del ListBox, possiamo accedere all'elemento selezionato nella lista, per questo se l'elemento selezionato è diverso dalla stringa vuota si può procedere con l'esecuzione del codice in caso contrario la procedura termina senza nessun effetto. All'interno del blocco *If..Then..End If* viene valorizzato l'oggetto *ObjFilmcon* un riferimento all'elemento della collezione che ha come chiave il titolo selezionato nel ListBox e ne vengono mostrati i dati.

LA PROCEDURA VISUALIZZADATI

La procedura *VisualizzaDati* deve visualizzare nei TextBox corrispondenti, i dati del film selezionato:

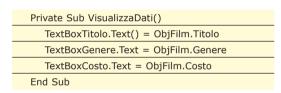
```
Private Sub VisualizzaDati()
    TextBoxTitolo.Text() = ObjFilm.Titolo
    TextBoxGenere.Text = ObjFilm.Genere
    TextBoxCosto.Text = ObjFilm.Costo
End Sub
```

Vi lascio, infine, il compito di scrivere il codice associato al tasto *Cancella* che dovrà cancellare un elemento dalla collezione e dal ListBox, confrontatelo poi con il codice riportato nel box

Fig. 1: *L'applicazione Videoteca in esecuzione*

CONCLUSIONI

Se vi piace un software ben progettato, sarete certamente d'accordo che la programmazione ad oggetti è una tecnologia attraente. Parlando di VB.NET utilizzeremo spesso i concetti e la terminologia usata in questa serie, poiché tutti i controlli in sono degli oggetti, e molti di essi presentano al loro interno una collezione di oggetti.

Luigi Buono

DISTRUTTORI

Il distruttore è una porzione di codice per il rilascio delle risorse di sistema eseguito ogni volta che si rilascia il riferimento ad un oggetto. Tipiche operazioni di rilascio delle risorse di sistema sono: la chiusura di un file, la chiusura di una connessione a database o il salvataggio di informazioni di stato. Prima di distruggere un oggetto, il compilatore chiama automaticamente il metodo Finalize per oggetti che definiscono una routine Sub Finalize.

Costruire user control con ASP.NET

I custom control consentono di creare oggetti che favoriscono il riutilizzo del codice e facilitano la scrittura delle applicazioni. È una tecnica basilare per creare software dai lavori precedenti

Il riutilizzo del codice è uno dei fondamenti su cui l'object oriented programming si basa. Poter riutilizzare qualcosa in più punti dell'applicazione, specie in ambito web, consente di risparmiare tempo, perché una volta scritto il codice, si può riutilizzare ed impiegare quel tempo per qualcosa di meglio. In secondo luogo, consente di scrivere meno codice, con il conseguente vantaggio di esser certi di introdurre meno bug e soprattutto di potersi concentrare in un solo punto qualora siano necessarie modifiche, anche sostanziali. Infine, centralizzare in un solo punto le funzionalità, diventa anche un vantaggio in fase di manutenzione ordinaria, perché è più facile trovare il posto in cui una data funzionalità è stata inserita.

RIUTILIZZARE IL CODICE CON ASP.NET

ASP.NET consente due modalità principali per riutilizzare il codice, dove per questo si intenderà, d'ora in poi, una certa funzionalità ed il codice (X)HTML associato:

- **Custom control**: sono classi vere e proprie, che ereditano da Control (o da un altro controllo di ASP.NET) e sono principalmente composte da codice C# (o VB.NET);
- **User control**: sono pezzi di pagina, composti più da HTML che da codice vero e proprio.

I vantaggi dell'uno o dell'altro approccio sono evidenti e consistono principalmente nell'essere utili in scenari diversi. I *custom controls*, per esempio, sono la scelta obbligata quando un *control* è molto complesso oppure, più semplicemente, va distribuito su più progetti. Gli user control, dal canto loro, sono molto immediati e si realizzano molto semplicemente, ma sono sicuramente meno portabili, su più progetti, di un *custom control*. Infine, mentre i primi sono compilati in *class libray*, i secondi sono dei file con HTML e codice mischiato, dall'estensione *.ascx.*

Concettualmente, dunque, gli user control ricordano molto da vicino gli *include*, utilizzati ad esempio con *Classic ASP* per la costruzione dei template, piuttosto che per includere all'interno di più pagine le stesse funzionalità. In realtà, a differenza di un include, uno *user control* è un oggetto a tutti gli effetti e dunque può essere programmato, prima di tutto.

Questo vuol dire, molto semplicemente, che si hanno a disposizione proprietà, metodi ed eventi all'interno di questo oggetto, e che è possibile dal contenitore (che in genere è la pagina) modificarne lo stato.

ANATOMIA DI UNO USER CONTROL

Per capire al meglio come sfruttare uno *user control,* è necessario capirne prima di tutto la struttura. Per prima cosa, possiamo vederlo come una normale pagina. La differenza è che in realtà, a differenza della pagina, è solamente un pezzo di questa, e non è detto che sia autosufficiente, in quanto a funzionalità. Se provate a creare uno *user control* con VS.NET, noterete che rispetto alla pagina, la classe da cui eredita è *UserControl,* contenuta sempre nel namespace *System. Web.UI*. È tuttavia possibile scrivere, proprio come nel caso delle pagine, oggetti che contengano il codice direttamente nel corpo, utilizzando la tecnica nota come code inline.

Dunque, proprio come in una pagina, abbiamo a disposizione gli eventi della stessa, come *Page_Load*. In più, uno user control è inserito nella pagina proprio come un normale *server control,* attraverso una sintassi del tipo:

```
<ioprogrammo:news runat="server" id="news" />
```

Evidentemente c'è qualcos'altro dietro che al momento ignoriamo e che in seguito lo affronteremo. Ma come si può notare, non si tratta di un semplice include, bensì di un oggetto vero e proprio che quindi possiamo creare, in modo da essere completamente programmabile dall'oggetto contenitore.

IL NOSTRO PRIMO USER CONTROL

Per capire meglio la potenza e la flessibilità che questi oggetti offrono allo sviluppatore, vediamo come crearne uno semplice, che ci consenta di toccarne con mano i vantaggi. Per questo primo test creeremo un semplice user control che mostra a video, stampata, la data attuale. Per prima cosa, dunque, definiamone la struttura:

```
<%@Control Language="C#"%>
<p>Data attuale: <asp:literal id="data"
                              runat="server" /></p>
```

Successivamente aggiungiamone il codice:

```
void Page_Load()
{data.Text = DateTime.Now.ToLongDateString();}
```

Creiamo una semplice pagina, ed inseriamo in cima alla stessa:

```
<%@ Register TagPrefix="ioprogrammo"
             TagName="Data" Src="test1.ascx" %>
```

Quindi nel punto in cui vogliamo che la data venga inserita:

```
<ioprogrammo:Data runat="server" />
```

Possiamo anche includerne più di uno per pagina, senza dover replicare il codice. Tra l'altro, il vantaggio di usare gli *user control* è che possono essere anche scritti in un linguaggio differente da quello della pagina, ammesso che questo abbia senso per quanto riguarda la coerenza dell'applicazione.

REGISTRARE UNO USER CONTROL

L'abbiamo appena fatto in maniere empirica, ma vale la pena soffermarsi un attimo sul funzionamento della registrazione. Nell'esempio precedente abbiamo inserito una riga in alto, nella pagina contenitore. È un'istruzione obbligatoria che serve a dire al parser di ASP.NET dove andare a recuperare il codice sorgente dello user control. Si tratta di un codice che ha questa forma:

```
<%@ Register TagPrefix="prefix" TagName="name"
                              Src="file.ascx" %>
```

Ed i cui valori hanno esattamente questo significato quando andiamo ad inserire il *control* nella pagina:

```
<prefix:name runat="server" />
```

L'attributo *Src* è invece riferito al path per arrivare al file sorgente e può essere un percorso relativo piuttosto che uno assoluto, ma ovviamente deve sempre puntare ad una risorsa raggiungibile localmente. La prima riga va inserita ogni volta che dobbiamo usare uno user control all'interno di una pagina ed è tutto quello di cui abbiamo bisogno per poter poi inserire il control.

COMUNICARE CON LA PAGINA

Creato il primo *control,* dobbiamo ovviamente porci il problema di come farlo comunicare con il resto del mondo, che per noi sono gli altri soggetti presenti sulla pagina. Uno *user control,* infatti, può essere contenuto all'interno di una pagina ma anche all'interno di un altro *user control,* dunque di fatto non è detto che l'oggetto contenitore sia a propria volta un altro *control* di questo tipo, anzi spesso è proprio così. Dunque in questo frangente, per semplicità, ci riferiremo sempre all'argomento specificando come oggetto contenitore una pagina, ma i concetti sono del tutto identici qualora il contenitore sia un altro *user control.* L'esempio che abbiamo appena visto, nella sua banalità, ci permette comunque di capire che, in fin dei conti, uno user control è un qualsiasi oggetto e quindi pertanto può essere esteso. A questo proposito, vediamo come aggiungere al nostro control una serie di proprietà per far sì che l'aspetto grafico dello stesso sia più gradevole. Cominciamo con il dotare il nostro oggetto di una proprietà di testo di nome *CssClass,* che useremo per specificare uno stile da associare al testo scritto:

```
// proprietà per specificare uno stile
private string cssClass;
public string CssClass
{ get {return cssClass;}
  set {cssClass = value;}
}
```

I TUOI APPUNTI

Utilizza questo spazio per le tue annotazioni

L'AUTORE

Daniele Bochicchio è il content manager di *ASPItalia.com*, community che si occupa di ASP.NET, Classic ASP e Windows Server System. Il suo lavoro è principalmente di consulenza e formazione, specie su ASP.NET, e scrive per diverse riviste e siti. È Microsoft ASP.NET MVP, un riconoscimento per il suo impegno a supporto delle community e per l'esperienza maturata negli anni. Il suo blog è all'indirizzo http://blogs.aspitalia.com/daniele/

Continuiamo le modifiche associando questo valore alla rispettiva proprietà di un panel all'interno del quale avremo rinchiuso il testo già presente sul *control*. Ovviamente dovremo specificare, nella pagina, il valore da associare a questa proprietà. Possiamo farlo semplicemente in questo modo:

```
<ioprogrammo:Data id="data1" runat="server"
                                    CssClass="date"/>
```

Facciamo la stessa modifica per poter specificare un testo alternativo, da associare ad un *control Literal* inserito sul controllo, da visualizzare prima della data:

```
// proprietà per specificare un testo di descrizione
private string text = "Data attuale:";
public string Text
{  get {return text;}
   set {text = value;}
}
```

Come si può notare, in questo caso abbiamo associato un valore di default alla variabile privata che internamente la nostra classe utilizza per la persistenza del dato. È una pratica molto diffusa quando si realizzano *controls*, perché consente di rendere meno impegnativa la definizione delle proprietà necessarie al funzionamento del control, ma al tempo stesso consente di arrivare ad avere una personalizzazione completa. Conclude il nostro control la definizione di un evento sul *Page_Load*, che vada a formattare gli oggetti della pagina con le proprietà specificate:

```
void Page_Load()
{  testo.Text = Text;
   data.Text = DateTime.Now.ToLongDateString();
   if (cssClass != null)
       contenitore.CssClass = cssClass;
}
```

MANTENERE LO STATO: IL VIEWSTATE

Provate ad assegnare le proprietà attraverso codice e noterete che, tra i vari *PostBack*, i valori non vengono mantenuti e che, soprattutto, gli eventi si verificano, nella pagina e nei controls, secondo modalità differenti. Quest'ultimo aspetto è approfondito nel box laterale. Per quanto riguarda il primo problema, è essenzialmente dovuto al fatto che tutti i controls che usiamo sulle pagine mantengono il loro stato attraverso il *ViewState*, di cui abbiamo già parlato in uno dei precedenti articoli. Dunque per fare in modo che il nostro control mantenga lo stato delle proprietà tra i vari *PostBack*, dovremo fare una modifica che consenta di salvare e leggere il valore di queste proprietà da questo *state bag*. La proprietà *CssClass*, usata per specificare uno stile, diventa:

```
// proprietà per specificare uno stile
private string cssClass;
public string CssClass
{  // leggiamo dal ViewState
   get {return ViewState["data_cssClass"] as String;}
   // scriviamo nel ViewState
   set {
      ViewState["data_cssClass"] = value;
      cssClass = value; }
}
```

E la proprietà *Text* segue la stessa strada appena spiegata, ovviamente. Come si può notare, nel recuperare la proprietà andiamo in realtà a leggere direttamente dal *ViewState*, dato che in fase di assegnazione ne abbiamo salvato il valore.
L'ultima modifica poi, riguarda il momento in cui applicare le proprietà al control. Tra gli eventi a disposizione per chi sviluppa *controls*, quello che sicuramente rappresenta il momento migliore in cui andare ad applicare eventuali proprietà di tipo *"grafico"* è *Page_PreRender*, che si verifica dopo *Page_Init* e *Page_Load* e prima che la pagina venga costruita per essere inviata al client. Dunque sposteremo tutta la logica in questo evento, in modo che sia possibile effettuare tutte le operazioni quando il *ViewState* è stato già caricato. A questo proposito, andremo ad aggiungere un pulsante nella pagina ASP.NET alla cui pressione lo stile associato al nostro *user control* sarà modificato:

```
void cambia(Object o, EventArgs e)
{  Trace.Write(data1.CssClass);
   if (data1.CssClass == "date")
      data1.CssClass = "datedark";
   else
       data1.CssClass = "date";
}
```

Dopo questa ultima modifica siamo pronti per poter costruire control decisamente più avanzati.

UNO USER CONTROL PER L'ACCESSO AI DATI

Uno scenario in cui spesso uno user control può tornare utile è la definizione di una griglia per la visualizzazione dei dati, che includa funzionalità che sono ripetute più di una volta all'interno della nostra applicazione. In particolare, l'esempio

che prenderemo in considerazione si basa su uno *user control* contenente un *DataGrid* che mostra dati prelevati da un database, come abbiamo imparato a fare nel numero precedente. In più in questo caso abbiamo aggiunto la possibilità di visualizzare in automatico un messaggio qualora l'estrazione non dovesse produrre risultati, caratteristica molto utile ma che di default il *DataGrid* non fornisce. Partiamo dunque con il creare lo *user control,* definendo due proprietà pubbliche, una di nome *DataSource* per specificare la sorgente dati, ed una di nome *NoRecordText* per associare un testo alternativo al messaggio che appare quando non ci sono record:

```
// proprietà per specificare un DataSource
private object dataSource;
public Object DataSource
{  // leggiamo dal ViewState
   get {return ViewState["data_dataSource"];}
   // scriviamo nel ViewState
   set {  ViewState["data_datasSource"] = value;
          dataSource = value; }
}
```

Dobbiamo usare come tipo *Object* perché è possibile fare il binding ad una *DataGrid* di un numero molto elevato di oggetti, inclusi quelli personalizzati. Inseriamo quindi nel control un *DataGrid* con ID *"dg"* ed inseriamo nell'evento *Page_PreRender* il codice necessario ad associare la fonte dati al nostro *DataGrid* ed al tempo stesso a visualizzare un messaggio di errore in caso di mancanza di record:

```
void Page_PreRender()
{  bool hasRows = false;
    // associo il dataSource al controllo
    if (dataSource != null)
    {  dg.DataSource = dataSource;
        dg.DataBind();
        // ci sono record?
        hasRows = (dg.Items.Count>0); }
   if (!hasRows)
   {   // avvisiamo che non ci sono record
        this.Controls.Add(new LiteralControl(
                                        noRecordText));
        // nascondiamo controllo
        dg.Visible = false; }
}
```

Nella pagina andiamo a registrare il control appena creato, lo inseriamo all'interno della web form insieme ad un pulsante ed effettuiamo il binding dei dati attraverso questa funzione:

```
// carico i dati
void bindData(string filtro)
{ [...]
   // creo il dataset e lo riempio con i dati
   DataSet ds = new DataSet();
   query.Fill(ds);
   // associo i dati al nostro user control
   dg.DataSource = ds;
}
```

Al pulsante invece assoceremo questo event handler:

```
void filtro(Object o, EventArgs e)
{  // filtro la griglia
    bindData("WHERE title_id= '1'");
}
```

Come si può notare dalle immagini, nel primo caso verrà mostrata la griglia, nel secondo provocando un'associazione che non restituisce record, verrà visualizzato il nostro messaggio.

CONCLUSIONI

Costruire oggetti programmabili e riutilizzabili con ASP.NET è davvero molto semplice. L'argomento user control può sembrare banale, ma in realtà è uno dei punti di forza che un'applicazione web è in grado di sfruttare al meglio, poiché consente di riutilizzare molto codice, ma soprattutto di modellare gli oggetti in base alle proprie necessità.

Daniele Bochicchio

LA SEQUENZA DEGLI EVENTI TRA PAGINA E CONTROLS

Sia la pagina, che eredita dalla classe Page, sia gli user controls, che ereditano invece da UserControl, hanno una serie di eventi in comune che servono per eseguire codice in base al verificarsi di determinate condizioni. È importante conoscere, in applicazioni complesse, il giusto ordine in cui si verificano questi eventi all'interno dei controls. In tutto ci sono 9 eventi, tra cui sicuramente i più utilizzati, in ordine di successione, sono:

- ***Page_Init***
- ***Page_Load***
- ***Page_PreRender***
- ***Page_Render***

Se una pagina contiene uno user control, quest'ultimo avrà la stessa sequenza di eventi della pagina, con la differenza che ogni evento sarà invocato subito dopo l'evento corrispondente contenuto nella pagina. In pratica, la sequenza sarà la seguente:

- **Page_Init**
  - *** Della pagina**
    - **- Degli user Controls**
- **Page_Load**
  - *** Della pagina**
    - **- Degli user Controls**
- **Page_PreRender**
  - *** Della pagina**
    - **- Degli user Controls**
- **Page_Render**
  - *** Della pagina**
    - **- Degli user Controls**

Se ad esempio dobbiamo passare, da uno user control, informazioni che la pagina dovrà leggere nell'evento *Page_Load,* dovremo scrivere il codice all'interno del *Page_ Init.* Analogamente, se uno *user control* è contenuto in un altro user control, questi eventi si verificheranno sempre nel control contenitore.

L'URL che non ti fa "arrabbiare"

Utilizzare correttamente gli URL è l'unico metodo per scrivere applicazioni WEB in grado di gestire correttamente le informazioni Ma cos'è un URL? Che c'entra JavaScript? Scopriamolo in questo articolo

Supponete di avere a disposizione un database di parole e di indirizzi internet. Voi siete proprietari di un sito internet un po' particolare. Il vostro sito propone una pagina che, dato un elenco alfabetico di parole, consente ad un utente di cliccare su una di queste e ottenere in risposta un elenco di siti che la contengono, con il relativo indirizzo. Per questioni di velocità non volete che la pagina sia ricaricata dopo ogni clic, ma utilizzate uno script JavaScript che scrive sulla pagina dinamicamente i link in corrispondenza dell'*OnClic*. Tutto chiaro?
Sembrerebbe tutto limpido come l'acqua di fonte, ed invece qui ci scappa l'inghippo! Vediamo quali sono le trappole che si nascondono dietro la stampa di URL a schermo tramite Javascript. Qualche semplice accorgimento ci aiuterà a non compiere errori.

UN ESEMPIO PRATICO

Consideriamo un URL formato in maniera un po particolare ad esempio:

```
http://www.google.it/search?hl=it&q=javascript&btnG
                              =Cerca+con+Google&meta=
```

Se stampate a video questo URL tramite il seguente codice JavaScript

```
var strStringaDaCercare = "javascript",
str = "<a href=\"";
str += encodeURI(
            "http://www.google.it/search?hl=it&q="
                          + strStringaDaCercare);
str +=  "\">Ricerca parola: " + strStringaDaCercare
                                        + "</a>";
document.write (str);
```

Vi verrà presentato un classico URL da cliccare. Se cliccate, si aprirà una pagina di "Google" con i risultati di una ricerca per la parola *"Javascript"*. Ora non è il caso di entrare nel dettaglio dei meccanismi di Google ma in ogni caso nella formattazione del Link notate che dopo il parametro *"q"* c'è la parola *"Javascript"*. Richiamando l'Url di Google con dei parametri opportuni si ottiene in restituzione la pagina in questione. Non è su questo punto che ci vogliamo concentrare, ma sui caratteri che vengono utilizzati nella generazione del Link, quindi smettetela di concentrarvi su Google e passiamo oltre.
Non si sa mai cosa un utente potrebbe cercare, quindi supponiamo che invece della parola *"Javascript"* vogliate ricercare la frase *"j@vascript è bello ? (ma sì che è bello!)"*. Cambiamo allora il valore della variabile *strStringaDaCercare* ottenendo il codice seguente:

```
var strStringaDaCercare = "j@vascript è bello ?
                              (ma sì che è bello!)";
str = "<a href=\"";
str += "http://www.google.it/search?hl=it&q=" +
                                strStringaDaCercare;
str +=  "\">Ricerca parola: " + strStringaDaCercare
                                        + "</a>";
document.write (str);
```

Se clicchiamo sul link così generato, non otteniamo il risultato sperato. Google, invece di cercare la stringa che gli abbiamo passato cercherà la seguente: *"j@vascript ? bello ? (ma s? che ? bello!)"* in cui i caratteri accentati sono stati sostituiti da *?* (inseriti da Google per significare che il carattere "è" non è riconosciuto). Si sarà intuito che ovviamente l'errore non è di Google, ma è nostro, in quanto non abbiamo codificato il modo opportuno la

stringa inviata.
Proviamo allora il seguente codice:

```
var strStringaDaCercare = "j@vascript è bello ? (ma
                                           sì che è bello!)";
str = "<a href=\"";
str += encodeURI("http://www.google.it
            /search?hl=it&q=" + strStringaDaCercare);
str +=  "\">Ricerca parola: " + strStringaDaCercare
                                              + "</a>";
document.write (str);
```

Questo spezzone di codice ha di diverso dal precedente il fatto che alla seconda linea è stato inserito la funzione *encodeURI( )* che codifica correttamente tutto il percorso dell'ancora (che è a tutti gli effetti un indirizzo URI). Se ora proviamo il link così generato, abbiamo il risultato voluto (al di là del fatto che non si otterrà nulla nella pagina di ricerca, vista la stringa un po' assurda che si sta cercando...). Trucco svelato, articolo finito, andiamo tutti a provare il nostro giochino nuovo! E invece no, proviamo questa frase: *"Paolo&Rossi Associated"*.
Il codice sarà il seguente:

```
var strStringaDaCercare = "Paolo&Rossi Associated";

str = "<a href=\"";
str += encodeURI("http://www.google.it
            /search?hl=it&q=" + strStringaDaCercare);
str +=  "\">Ricerca parola: " + strStringaDaCercare
                                              + "</a>";
document.write (str);
```

Cliccando sul link, invece di trovarci la pagina dei risultati per la stringa *"Paolo&Rossi Associated"*, troviamo i risultati della ricerca della sola parola Paolo. Dov'è che sbagliamo? Il fatto è che la stringa ricercata contiene il carattere *"&"* tra il nome *'Paolo'* ed il nome *'Rossi'* che ha un significato particolare per gli indirizzi URI (quello di concatenare le variabili passate in *GET*). Quindi non viene decodificato.
Il codice corretto è il seguente:

```
var strStringaDaCercare = "Paolo&Rossi Associated";
str = "<a href=\"";
str += "http://www.google.it/search?hl=it&q="
        + encodeURIComponent(strStringaDaCercare);
str +=  "\">Ricerca parola: " + strStringaDaCercare
                                              + "</a>";
document.write (str);
```

Il valore contenuto in *strStringaDaCercare* deve essere considerato un componente dell'URI e quindi tutti i caratteri in esso contenuti vanno codificati utilizzando il metodo *encodeURIComponent( )* che non permette dei caratteri, come *"&"*, che sono permessi invece da *encodeURI( )*. In articoli successivi, parlando di passaggio di dati mediante javascript tra form HTML approfondiremo e generalizzeremo quanto discusso nel caso semplice di un unico parametro di ricerca (la cosa è leggermente più complessa...). Per adesso ricordatevi di fare sempre attenzione a usare correttamente i metodi, a secondo se si parla di URI o di sue componenti.

COSA BOLLE SOTTO UN URI

Fin qui abbiamo spesso usato la parola URI. In letteratura URI vuol dire *Uniform Resource Identifier.* Si intende per URI un puntatore che identifica in modo univoco una risorsa presente sulla rete. Un link è sicuramente un URI. Ora, il basamento di Internet è che tutti coloro che utilizzano questo mezzo come forma di diffusione dei contenuti o per un qualunque motivo, si devono adeguare a degli standard certi, di modo che ciascuno sia in grado di accedere ad internet in modo uniforme. Immaginate cosa potrebbe succedere se il 10% dei siti internet identificassero i link nella forma *"http((pippo;it))"*, invece che nella forma *"http://pippo.it"* a cui siamo abituati, e immaginate cosa succede se ognuno utilizzasse delle proprie regole per scrivere i link. Perciò si definiscono degli standard a cui tutti devono adeguarsi. Non si discosta da questa regola la definizione di un URI. La *RFC 3986* sancisce che un URI deve essere così composto:

```
scheme":" hier-parm ["?" query ]["#" fragmen]
```

dove a sua volta *hier-parm* è composto di

```
//Authorithy
/Path-absolute
/Path-rootless
/Path-Empty
```

Per farla breve, se

```
scheme->http, authorithy-> www.google.it, query->
                                         hl=it&q=javascript
```

Il nostro URI avrebbe la forma

```
http://www.google.it?hl=it&q=javascript
```

Ora, se invece della parola javascript volessi-

NOTA

UN PO' DI MATEMATICA...

In termini matematici si potrebbe dire l'insieme è invariante per trasformazioni f: K -> K dove f è uno dei nostri metodi encodeURI(), encodeURIComponent( ) oppure escape(), mentre K è l'insieme di caratteri che viene trasformato in se stesso. Ogni carattere che viene trasformato in se stesso, per un particolare metodo, è quindi un elemento dell'insieme invariante K di quel metodo. Nel seguito dell'articolo utilizzeremo il simbolo U nel senso insiemistica di unione di insiemi. La stringa: [A..Z]U[a..z]U[0..9]U[* + - . / @ _]
Sta quindi a significare l'insieme formato da tutti gli elementi [A..Z] più gli elementi [a..z] più ancora gli elementi [0..9] e [* + - . / @ _].

mo cercare la frase *"Java è un linguaggio?"* la nostra query assomiglierebbe a qualcosa del genere:

```
http://www.google.it?hl=it&qJava è un linguaggio?
```

Va da sé che questa formattazione non è corretta; primo perché ci sono degli spazi nella stringa da ricercare; secondo perché compare un punto interrogativo, al termine della frase che va in conflitto con la definizione di URI che prevede il punto interrogativo come simbolo per iniziare una query. Si evince da tutto ciò che esistono caratteri particolari che devono essere codificati prima di poter comparire all'interno di un URI. Ad esempio la codifica dello spazio bianco è *%20* Esistono tre metodi con cui si può effettuare la codifica. E ciascuno dei tre metodi stabilisce in che modo debba avvenire la codifica. Riassumendo, nella **Tabella 1** per ogni metodo di codifica, è riportato l'insieme di caratteri invariante sui quali il metodo non effettua alcuna codifica.

Metodo (f)	Insieme invariante (K)
encodeURI	[A..Z]U[a..z]U[0..9]U[! # $ & ' ( ) * + , - . / : ; = ? @ _ ~]
encodeURIComponent	[A..Z]U[a..z]U[0..9]U[! ' ( ) * - . _ ~]
escape(entità)	[A..Z]U[a..z]U[0..9]U[* + - . / @ _]

Tabella 1:

Utilizza questo spazio per le tue annotazioni

UN PO' DI BUON VECCHIO CODICE

Vediamo ora alcuni esempi pratici di applicazione di questi metodi, verificando proprio l'invarianza di questi insiemi di caratteri. Definiamo le seguenti stringhe:

```
// Definizione delle stringhe invarianti

strINVComune = "ABCDEFGHIJKL MNOPQRSTUVWXYZ
              abcdefghijklmnopqrstuvwxyz0123456789";
strINVencodeURI = "!#$&'()*+,-./:;=?@_~";
strINVencodeURICmp = "!'()*-._~";
strINVescape = "*+-./@ _";
```

Per prima cosa verifichiamo che l'insieme di caratteri *[A..Z] U [a..z] U [0..9]* è un insieme invariante per trasformazione rispetto ai tre metodi. Il codice seguente:

```
document.write("encodeURI(strINVComune) = " +
              encodeURI(strINVComune) + "<br>");
document.write("encodeURIComponent(strINVComune)
  = " + encodeURIComponent(strINVComune) + "<br>");
document.write("escape(strINVComune) = "
              + escape(strINVComune) + "<br>");
```

In output restituisce:

```
encodeURI(strINVComune) = ABCDEFGHIJKLMNOPQRS
  TUVWXYZabcdefghijklmnopqrstuvwxyz0123456789

encodeURIComponent(strINVComune) = ABCDEFGHIJK
               LMNOPQRSTUVWXYZabcdefghijklmnopq
                           rstuvwxyz0123456789

escape(strINVComune) = ABCDEFGHIJKLMNOPQRS
  TUVWXYZabcdefghijklmnopqrstuvwxyz0123456789
```

Rispetto alle stringhe *[A-Z],[a-z],[0-9]* i tre metodi si comportano allo stesso modo.
Verifichiamo ora come si comportano questi tre diversi metodi sui tre gruppi di caratteri invarianti. Per quanto riguarda il metodo: *encodeURI( )* il codice:

```
document.write("encodeURI(strINVencodeURI) = " +
              encodeURI(strINVencodeURI) + "<br>");
document.write("encodeURI(strINVencodeURICmp) =
    " + encodeURI(strINVencodeURICmp) + "<br>");
document.write("encodeURI(strINVescape) = " +
              encodeURI(strINVescape) + "<br>");
```

in output restituisce:

```
encodeURI(strINVencodeURI) = !#$&'()*+,-./:;=?@_~
encodeURI(strINVencodeURICmp) = !'()*-._~
encodeURI(strINVescape) = *+-./@_
```

Il risultato mostra proprio quanto ci si aspettava, e cioè che il metodo *encodeURI( )* che possiede l'insieme invariante composto dal maggior numero di caratteri rispetto agli altri insiemi, non codifica nessuno dei caratteri degli altri insiemi. In pratica, l'insieme invariante per il metodo *encodeURI( )* contiene i rimanenti due insiemi invarianti rispettivamente per i metodi *encodeURIComponent( )* ed *escape( )*, che ne sono quindi sottoinsiemi. Per quanto riguarda il metodo: *encodeURIComponent( )* il codice:

```
document.write("encodeURIComponent(strINVencode
                  URI) = " + encodeURIComponent(
                  strINVencodeURI) + "<br>");
document.write("encodeURIComponent(
  strINVencodeURICmp) = " + encodeURIComponent(
                  strINVencodeURICmp) + "<br>");
document.write("encodeURIComponent(strINVescape)
       = " + encodeURIComponent(strINVescape)
                                  + "<br>");
```

in output restituisce:

```
encodeURIComponent(strINVencodeURI) =
```

```
!%23%24%26'()*%2B%2C-
.%2F%3A%3B%3D%3F%40_~
encodeURIComponent(strINVencodeURICmp) = !'()*-._~
encodeURIComponent(strINVescape) =
*%2B-.%2F%40_
```

Vediamo quindi che l'unico insieme di caratteri invariante è *[!'()*-._~]*, ossia quello che abbiamo in precedenza definito essere invariante per il metodo *encodeURIComponent( )*.
In modo del tutto analogo, per il metodo *escape ( )* il codice:

```
document.write("escape(strINVencodeURI) = " +
escape(strINVencodeURI) + "<br>");
document.write("escape(strINVencodeURICmp) = " +
escape(strINVencodeURICmp) + "<br>");
document.write("escape(strINVescape) = " +
escape(strINVescape) + "<br>");
```

produce in output:

```
escape(strINVencodeURI) =
%21%23%24%26%27%28%29*+%2C-
./%3A%3B%3D%3F@_%7E
escape(strINVencodeURICmp) = %21%27%28%29
*-._%7E
escape(strINVescape) = *+-./@_
```

Anche per questo metodo valgono le considerazioni fatte per *encodeURIComponent( )* e verifichiamo quindi che l'unico insieme di caratteri invariante è *[* + -./@ _]*, ossia quello che abbiamo in precedenza definito essere invariante per il metodo *escape( )*. La tabella completa delle codifiche per tutti i caratteri ASCII compresi tra 32 e 126 è presente nel codice allegato al presente articolo dove è possibile trovare lo script per il calcolo di questa tabella. Su una generica stringa, i tre metodi forniranno in output stringhe differenti, come nel seguente esempio di codice:

```
var strEsempio = "j@vascript è bello ?
(ma sì che è bello!)";
document.write("strEsempio = " + strEsempio
+ "<br>");
document.write("encodeURI(strEsempio) = "
+ encodeURI(strEsempio) + "<br>");
document.write("encodeURIComponent(strEsempio)
= " + encodeURIComponent(strEsempio) + "<br>");
document.write("escape(strEsempio) = "
+ escape(strEsempio) + "<br>")
```

nel quale i tre metodi restituiscono come output tre stringhe diverse:

```
strEsempio = j@vascript è bello ? (ma sì che è bello!)
encodeURI(strEsempio) =
j@vascript%20%C3%A8%20bello%20?%20(
ma%20s%C3%AC%20che%20%C3%A8%20bello!)
encodeURIComponent(strEsempio) =
j%40vascript%20%C3%A8%20bello%20%3F%20
(ma%20s%C3%AC%20che%20%C3%A8%20bello!)
escape(strEsempio) = j@vascript%20%E8%
20bello%20%3F%20%28ma%20s%EC%20che%
20%E8%20bello%21%29
```

È importante ricordare che nella manipolazione delle stringhe URI o delle loro componenti non si deve mai fare uso del metodo *escape( )*.
Per quanto riguarda i metodi *decodeURI(), decodeURIComponent( )* ed *unescape( )*, tali metodi eseguono esattamente l'operazione inversa dei corrispettivi *encodeURI( ), encodeURIComponent( )* ed *escape ( )*.
L'esempio di codice:

```
strStringa= "j@vascript è bello ? (ma sì che è bello!)";
document.write("decodeURI(encodeURI(strStringa)) =
" + decodeURI(encodeURI(strStringa)) + "<br>");
document.write("decodeURIComponent(encodeURICo
mponent(strStringa)) = " + decodeURIComponent(
encodeURIComponent(strStringa)) + "<br>");
document.write("unescape(escape(strStringa)) = "
+ unescape(escape(strStringa)) + "<br>");
```

restituisce per tutti e tre i casi il valore della stringa *strStringa* di partenza.

CONCLUSIONI

La corretta gestione delle URI ci consente di scrivere pagine WEB efficaci quando l'output si presenta in modo molto complesso. D'altra parte i link sono le fondamenta del web.

Danilo Berta

METODI ESCAPE ED UNESCAPE...

I metodi escape ed unescape sono stati deprecato dall'ECMA, ma lo descriveremo perché ancora utilizzato in casi particolari. Per quanto riguarda le possibilità di utilizzo dei metodi, questi non devono essere utilizzati per la gestione delle URI, cosa che si faceva un po' di tempo fa. Personalmente non riesco a trovare nessun esempio dell'utilizzo di questo metodo, a meno che non si pensi a qualche script CGI un po' datato che utilizza unescape( ) per decodificare le stringhe e quindi si aspetta di ricevere in input stringhe codificate con escape(). In definitiva, se avete bisogno di usarlo, qualcuno vi dirà di usarlo...

DEFINIZIONE DI URI...

**Il termine URI è un acronimo che sta ad indicare: *Uniform Resource Identificator*, è in pratica una stringa di caratteri che, seguendo ben precise regole di sintassi, permette l'individuazione univoca di una risorsa su Internet. Facciamo notare che "una risorsa su Internet" non è solo l'indirizzo di una pagina web, definito da una URL *(Uniform Resource Locator)*, ma anche un indirizzo di posta elettronica, un indirizzo di news su *USENET*, etc.
Quindi il concetto di URI in qualche modo estende quello di URL.
La definizione dettagliata delle regole semantiche alle quali deve sottostare una URI sono descritte nella RFC del W3C denominata RFC2396 e raggiungibile al seguente indirizzo... URI... (http://www.w3.org/2002/11/dbooth-names /rfc2396-numbered_clean.htm).
Senza entrare troppo nel merito, il paragrafo 1.6 di questa "*Request for Comment*" definisce in modo rigoroso la struttura generale di una URI.**

Gestire i contatti con Symbian OS

Nella programmazione dei cellulari è importante capire come vengono gestite le applicazioni standard prima di avventurarci nella creazione di un nostro programma. Analizziamo il funzionamento della rubrica

Nei precedenti articoli abbiamo descritto la struttura generica di un programma per Symbian OS, più in particolare per la piattaforma Serie 60 di casa Nokia. Abbiamo anche parlato di come il framework grafico possa essere utilizzato per rappresentare le informazioni a video. A partire da questo articolo vedremo cosa Symbian OS ci offre nello specifico, trattando i singoli argomenti come dei mattoncini nella costruzione del software finale. Seguendo questo profilo, non possiamo non iniziare dalla gestione dei contatti che risiedono sul telefono cellulare. Come al solito, la parte teorica ci servirà per capire i meccanismi di funzionamento, e la parte pratica, tramite l'ausilio di piccoli esempi, ci aiuterà ad applicare le conoscenze acquisite.

CONTATTI E VCARDS

I contatti, sui moderni smartphone, non sono altro che un insieme di campi contenenti un'etichetta ed un valore. Ogni campo può rappresentare un diverso tipo di dato, come una stringa, un numero o una data. Inoltre il numero di campi è variabile, in modo da abbracciare le necessità di una molteplicità di utenti. Pensiamo, ad esempio, ad una persona con più nomi o più numeri da memorizzare. Poniamoci adesso in un caso reale. Cosa succederebbe se, cambiando il cellulare, dovessimo trasferire i contatti da un dispositivo all'altro? E se fosse un palmare?
Affinché i contatti possano essere trasferiti da un dispositivo all'altro senza problemi è necessario che tutte le periferiche memorizzino i contatti secondo uno standard preciso.
Tale standard si chiama *vCard (RFC 1521)*.
Una *vCard* esprime il modo in cui devono essere rappresentate le informazioni inerenti un contatto. Symbian OS gestisce i propri dati tramite le *vCard* in maniera trasparente per il programmatore e per l'utente. Ciò significa che l'accesso e la manipolazione delle informazioni è possibile solo tramite l'ausilio di opportune API, che esonerano dal comprendere "dove" e "in che modo" questi dati devono essere conservati.

IL DATABASE DEI CONTATTI

Dal punto di vista fisico il database standard dei contatti è contenuto interamente nel file *contacts.cdb* che risiede nella directory *\system\data*. Dal punto di vista logico, è possibile operare sul database tramite la classe *CContactDatabase*. Prima di ogni cosa è dun-

TIPOLOGIE DI CAMPO

È possibile trovare un elenco dettagliato di tutti i tipi di campo nel file cntdef.h presente ovviamente in ogni SDK. Tali costanti devono essere passate come secondo parametro nella costruzione dell'oggetto CContactItemField.

KUidContactFieldPhoneNumber
KuidContactFieldFamilyName
KuidContactFieldAdditionalName
KuidContactFieldEMail
KuidContactFieldSms
KuidContactFieldNote
KuidContactFieldBirthday
KuidContactFieldUrl
KuidContactFieldPicture
KuidContactFieldRingTone

Il file header sopra citato contiene anche le costanti per il mapping. Eccone alcune:

Nome	KUidContactFieldVCardMapUnusedN
Cognome	KUidContactFieldVCardMapUnusedN
Compagnia	KUidContactFieldVCarcMapORG
Occupazione	KUidContactFieldVCarcMapTITLE
Telefono	KUidContactFieldVCarcMapTEL
Email/internet	KUidContactFieldVCarcMapEMAILINTERNET

Conoscenze richieste
Basi di C++, basi di Symbian C++

Software
Visual C++ 6 o sup., ActivePerl, Nokia SDK 1.2 o sup.

Impegno

Tempo di realizzazione

que necessario creare l'oggetto attraverso il quale effettuare le operazioni. Ciò è possibile grazie alla chiamata *CContactDatabase::OpelL()* che restituisce un puntatore ad un oggetto della stessa classe. Il seguente codice illustra la procedura:

```
CcontactDatabase * database =
                    CContactDatabase::OpenL();
CleanupStack::PushL(database);
```

La chiamata *OpenL()* effettua l'apertura del database, identificato come un descrittore di file, passato come parametro. L'assenza di parametri implica l'apertura del database di default. Nei nostri esempi manipoleremo solo il database di default, tuttavia, la funzione *CreateL()* ci permette di creare un nuovo database in un qualsiasi punto del filesystem. L'uso del *CleanupStuck* è necessario in quanto la classe non possiede un metodo *Close()* e di default le risorse appartenenti all'oggetto vengono liberate solo quando su di esso viene effettuata una chiamata delete. Nel caso in cui il programma uscisse a causa di un errore, il sistema in automatico provvederebbe a effettuare il delete di tutti gli oggetti presenti nel *CleanupStuck*. Solo dopo aver effettuato le operazioni è possibile liberare le risorse.

```
TInt numeroContatti = database->CountL();
CleanupStuck::PopAndDestroy(database);
```

Nell'esempio viene semplicemente richiamata la funzione *CountL()* ottenendo così il numero complessivo di contatti presenti nel database. Viene quindi prelevato l'oggetto dal *CleanupStuck* e in seguito distrutto.
Un ruolo molto importante è rivestito dalle operazioni di compressione e recupero nell'ambito del database dei contatti. Dal momento che la memoria fisica non è subito liberata in seguito alla cancellazione di un contatto, è necessario effettuare una compattazione dei dati assimilabile alla deframmentazione. Essendo un'operazione che richiede del tempo è opportuno eseguirla solo quando realmente necessaria. Vediamo come:

```
if (database->CompressRequired())
{
        database->CompactL();
}
```

CompressRequired() analizza lo spazio libero e verifica l'effettiva necessità di una compattazione. Se invece ci troviamo nella circostanza di un database corrotto (con errori), possiamo provare ad effettuare un ripristino tramite il seguente codice:

```
if (database->IsDamaged()) {
        database->RecoverL();
}
```

AGGIUNGERE UN CONTATTO

L'aggiunta di un nuovo contatto si può schematizzare in tre passi. Assumeremo di avere precedentemente aperto il database di default.

1. Creazione di un oggetto *CContactItem*
2. Aggiunta delle informazioni al contatto
3. Aggiunta del contatto al database

I dati appartenenti al contatto, seguendo lo standard *vCard*, sono suddivisi per campi.
Ogni campo è rappresentato da una coppia etichetta-valore ed appartiene ad una determinata tipologia. Un campo relativo ad un numero telefonico deve poter essere distinto da un campo relativo ad una data, nonostante entrambi contengano numeri. In realtà, come vedremo, l'informazione è sempre rappresentata tramite stringhe di testo. Ciò che conta è la connotazione che vogliamo dare ai singoli campi. Per chiarire le idee diamo un'occhiata al codice:

```
_LIT (LabelNome,  "Nome");
_LIT(Nome, "IoProgrammo");
_LIT(LabelTelefono, "Tel Redazione");
_LIT(Telefono, "02831212");
_LIT(LabelEmail, "Email");
_LIT(Email, "servizioabbonati@edmaster.it");
CcontactDatabase * database =
                    CContactDatabase::OpenL();
CleanupStack::PushL(database);
CContactItem * contatto = CContactCard::NewLC();
CContactItemField * campo =
        CContactItemField::NewLC(KStorageTypeText,
                    KUidContactFieldFamilyName);
campo->SetMapping(
                KUidContactFieldVCardMapUnusedN);
campo->SetLabelL(LabelNome);
campo->TextStorage()->SetTextL(nome);
contatto->AddField(*campo);
CleanupStack::Pop();
campo = CContactItemField::NewLC(
   KStorageTypeText, KUidContactFieldPhoneNumber);
campo->SetMapping(KUidContactFieldVCardMapTEL);
```

Utilizza questo spazio per le tue annotazioni

```
campo->SetLabelL(LabelTelefono);
campo->TextStorage()->SetTextL(Telefono);
contatto->AddField(*campo);
CleanupStack::Pop();
campo = CContactItemField::NewLC(
             KStorageTypeText, KUidContactFieldEMail);
campo->SetMapping(
           KUidContactFieldVCardMapEMAILINTERNET);
campo->SetLabelL(LabelEmail);
campo->TextStorage()->SetTextL(Email);
contatto->AddField(*campo);
CleanupStack::Pop();
database->AddNewContactL(*contatto);
CleanupStack::PopAndDestroy(2);
```

Le due classi di riferimento sono *CContactItem* e *CContactItemField*. La chiamata *NewLC()* ci restituisce il puntatore ad un oggetto della classe.

Seguendo le convenzioni Symbian, tale oggetto è già inserito nel *CleanupStuck*. Alla fine di ogni blocco effettuiamo quindi la chiamata *Pop()*. La costruzione del campo richiede due parametri.

Notiamo che il primo di essi è sempre *KStorageTypeText*, proprio perché in genere si è soliti usare delle stringhe di testo. Il secondo caratterizza invece il campo secondo lo scopo per cui è stato inserito.

Nell'esempio utilizziamo un campo di tipo generico, uno di tipo telefono e uno di tipo email/internet. La funzione *SetMapping()* abilita il supporto *vCard* ed è sempre preferibile utilizzarla. Anche in questo caso dobbiamo specificare il tipo di mapping, relativo al tipo di campo. Un piccolo elenco di parametri è presente nel box illustrativo. Per finire, la funzione *SetLabelL()* si preoccupa di assegnare l'etichetta del campo, mentre *TextStorage()->SetTextL()* ne assegna il valore.

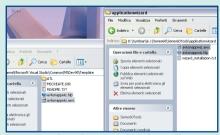
RICERCA E LETTURA DI UN CONTATTO

Ogni contatto nel database è identificato da un numero unico chiamato Id (identificativo). Per poter richiedere l'accesso ad un specifico contatto, e quindi ottenerne il puntatore all'oggetto, è necessario possedere il corrispondente Id. Il modo più semplice per ottenere l'identificativo di un contatto è scorrere il database effettuando una ricerca per campo. La rubrica di default, naturalmente, espone le informazioni per via grafica, utilizzando delle liste. Essa richiama semplicemente la funzione *Id()* sull'oggetto *CContactItem* selezionato, di cui si conosce il riferimento. Laddove quindi non sappiamo nulla sul database, dobbiamo effettuare un'iterazione in cerca di un qualcosa di specifico. Ecco il codice:

```
CContactDatabase * database =
                    CContactDatabase::OpenL();
CleanupStuck::PushL(database);
CContactItemFieldDef fieldDef = new (ELeave)
                          CContactItemFieldDef();
fieldDef->AppendL(KUidContactFieldFamilyName);
_LIT( Token, "IoProgrammo");
CidleFinder * finder = database->FindAsyncL(
                       Token, fieldDef, this);
```

Il punto chiave consiste nella chiamata alla funzione *FindAsyncL()*, ottimo servizio offerto dalla classe *CContactDatabase*. Tale funzione crea un elenco di Id relativi ai contatti che corrispondono ai parametri di ricerca. Il tipo di campo viene inserito all'interno di un oggetto *CContactItemFieldSet* e poi passato alla procedura di ricerca. È possibile anche inserire più tipi di campo richiamando più volte *AppendL()*. *CContactDatabase* offre diverse funzionalità di ricerca. Si è preferita una

CREAZIONE DI UN'APPLICAZIONE

1 **Installiamo gli strumenti che ci servono: Nokia SDK 1.2, Microsoft Visual C++ 6.0 e ActivePerl. Durante l'installazione di Visual C++ ricordiamoci di settare le variabili d'ambiente come in figura.**

2 **Aggiungiamo a Visual C++ il template *Nokia*. Copiamo *avkonappwiz.awx* e *avkonappwiz.hlp* da *Symbian\6.1\Series60\Series60Tools\applicationwizard* a *Microsoft Visual Studio \Common\MSDev98\Template***

3 **Apriamo Visual C++ e clicchiamo su *File*, poi su *New*. Dalla tab *projects* selezioniamo *Series 60 AppWizard*, digitiamo il nome del progetto, settiamo la directory. Nel dialog inseriamo il nome dell'applicazione.**

ricerca asincrona per migliorare le prestazioni. L'ultimo parametro, nel nostro caso this, deve essere un oggetto che implementa l'interfaccia *MIdleFindObserver,* contenente una funzione chiamata *IdleFindCallback()*. All'interno di tale funzione, richiamata ogni 16 contatti, inseriamo il codice che ci permetterà di manipolare l'informazione cercata.

```
void CIdleFindObserver::IdleFindCallback()
{
  if ( finder->IsComplete() )
  {
    if ( finder->Error() == KErrNone )
    {
      CContactIdArray * risultato = finder->
                                        TakeContactIds();
      for (TInt i=0; i<risultato->Count(); i++)
      {
        TInt cardId = (*risulato)[i];
        _LIT(message,"Contatto Trovato!");
        CAknConfirmationNote * note = new (ELeave)
                                   CAknConfirmationNote();
        note->ExecuteLD(message);
      }
      delete risultato;
    }
    delete finder;
    delete fieldDef;
  }
}
```

Nell'esempio, a ricerca ultimata, viene prelevato l'insieme di Id relativi ai contatti trovati e viene visualizzata una semplice informazione sul display. Un'interessante alternativa consiste nell'effettuare la ricerca per numero telefonico. Le API per i contatti si preoccupano persino di effettuare il matching su tutti i campi che siano stati catalogati come *KUidContactFieldPhoneNumber.* Tale lavoro è compiuto dalla funzione *PhoneMatchListL(),* ovviamente appartenente a *CContactDatabase.* Prima però è necessaria un'indicizzazione tramite le chiamate *InitLoadPhoneMatchesL()* e *LoadPhoneMatchesL()*. Vediamo il codice:

```
TInt itemRimasti = database->
                              InitLoadPhoneMatchesL();
while ( itemRimasti > 0) {
  itemRimasti = database->LoadPhoneMatchesL();
}
_LIT( telefono, "12345678");
CContactIdArray * idContatti = database->
                              PhoneMatchListL(telefono);
```

Dal momento che l'indicizzazione è asincrona dobbiamo utilizzare un ciclo *while* per aspettare il termine dell'operazione. La ricerca avviene inoltre sulle ultime otto cifre del numero di telefono.

APPLICAZIONI

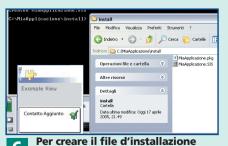
Avendo assimilato i concetti base, possiamo adesso manipolare i contatti del telefono direttamente all'interno dei nostri programmi. Tutto dipende ovviamente da ciò che vogliamo implementare. È possibile ad esempio scrivere un software che effettui un backup ogni sette giorni, oppure un album fotografico con il riferimento ai contatti, oppure un searcher avanzato che restituisca tutti i nomi delle persone che hanno un numero telefonico che inizia con XXXX (pensate quando sul dettaglio della bolletta non riuscite a riconoscere un numero perché mancano le ultime tre cifre).

Antonio Trapani

```
void CIdleFindObserver::IdleFindCallback() {
  if ( finder->IsComplete() ) {
    if ( finder->Error() == KErrNone ) {
      CContactIdArray * risultato = finder->TakeCo
      for (TInt i=0; i<risultato->Count(); i++) {
        TInt cardId = (*risulato)[i];
        _LIT(message,"Contatto Trovato!");
        CAknConfirmationNote * note = new (ELeave)
        note->ExecuteLD(message);
      }
      delete risultato;
    }
    delete finder;
    delete fieldDef;
  }
}
```

4 **Scriviamo il nostro codice aggiungendo funzioni alla struttura standard o creiamo nuove classi adatte alle nostre esigenze. Il codice proposto all'interno di questo articolo deve essere scritto a questo punto.**

```
C:\WINDOWS\System32\cmd.exe
C:\>cd MiaApplicazione\group
C:\MiaApplicazione\group>bldmake bldfiles
C:\MiaApplicazione\group>abld build thumb urel_
```

5 **Apriamo una console dei comandi e posizionamoci nella directory *group* appartenente al nostro progetto. Digitiamo *bldmake bldfiles* e subito dopo *abld build thumb urel.* Il programma è adesso compilato.**

6 **Per creare il file d'installazione portiamoci invece nella directory *install* e digitiamo *makesis HelloWorld .pkg*. Trasferiamo *HelloWorld.sis* sul telefonino, installiamolo, lanciamolo, e godiamoci i frutti del nostro lavoro.**

I trucchi del mestiere

Tips & Tricks

Questa rubrica raccoglie trucchi e piccoli pezzi di codice, frutto dell'esperienza di chi programma, che solitamente non trovano posto nei manuali. Alcuni di essi sono proposti dalla redazione, altri provengono da una ricerca su Internet, altri ancora ci giungono dai lettori. Chi volesse contribuire, potrà inviare i suoi Tips&Tricks preferiti. Una volta selezionati, saranno pubblicati nella rubrica. Il codice completo dei tips è presente nel CD allegato nella directory *tips*\\ o sul Web all'indirizzo: cdrom.ioprogrammo.it.

VISUAL BASIC

CONCATENARE STRINGHE IN VISUAL BASIC

Ovviamente si potrebbe utilizzare l'operatore &, tuttavia l'uso dell'operatore & fa si che se concatenate un certo numero di piccole stringhe in una più grande, viene allocata e deallocata la memoria ogni volta che l'operatore viene incontrato.
Un modo per ottimizzare la concatenazione è usare la seguente classe.

```
Private strText             As String
Public lngAllocated         As Long
Public lngUsed              As Long
Public lngAllocSize         As Long

Private Sub Class_Initialize()
   lngAllocSize = 1000
End Sub

Public Sub Add(strAddString As String)
   Dim lngLen               As Long
   Dim lngToAllocate        As Long

   lngLen = Len(strAddString)
   If lngLen > 0 Then
      If lngUsed + lngLen > lngAllocated Then
         lngToAllocate = lngAllocSize * (1 + (lngUsed +
                         lngLen - lngAllocated) \ lngAllocSize)
         strText = strText & String$(lngToAllocate, " ")
         lngAllocated = lngAllocated + lngToAllocate
      End If

      Mid$(strText, lngUsed + 1, lngLen) = strAddString
      lngUsed = lngUsed + lngLen
   End If
End Sub

Public Function GetStr() As String
   GetStr = Mid$(strText, 1, lngUsed)
End Function
```

PHP

COME CONTROLLARE SE UN FILE ESISTE IN PHP?

```
<?php
$filename = '/path/to/foo.txt';

if (file_exists($filename))
{
   print "Il file $filename esiste";
} else
{
   print "Il file $filename non esiste";
}
?>
```

COME VISUALIZZARE IN QUANTO TEMPO VIENE CARICATA UNA PAGINA PHP?

```
<?php
// Funzione: restituisce il tempo in microsecondi
function getmicrotime()
{
  list($usec, $sec) = explode(' ', microtime());
  return ((float)$usec + (float)$sec);
}
$tempo_inizio = getmicrotime();

//Non fare niente, 1000 volte
for ($i=0; $i<1000; ++$i);

//Si sottrae dal microtempo attuale quello iniziale
$tempo_esecuzione = getmicrotime() - $tempo_inizio;

//Otteniamo cosÃƒÂ¬ il tempo impiegato per eseguire le istruzioni
                                                    dello script
echo 'Tempo di esecuzione: ', $tempo_esecuzione , ' secondi';
?>
```

ASP.NET

COME ACCEDERE A UN DATABASE MYSQL?

```
<%@ import Namespace="System.Data" %>
<%@ import Namespace="System.Data.Odbc" %>
<%@ Page Language="C#" %>
<script runat="server">
public void Page_Load(Object sender, EventArgs e) {
 DataTable dtRecords = GetDataTable("SELECT * FROM newone");
 foreach(DataRow dr in dtRecords.Rows) {
 Response.Write(dr["FirstName"].ToString() + " "
                      + dr["LastName"].ToString() + "<br/>");} }
private static string GetConnection() {
 return "DRIVER={MySQL ODBC 3.51 Driver};
 Server=localhost;Database=testdatabase";}
public static DataTable GetDataTable(string sql) {
        DataTable rt = new DataTable();
        DataSet ds = new DataSet();
        OdbcDataAdapter da = new OdbcDataAdapter();
        OdbcConnection con = new
                          OdbcConnection(GetConnection());
        OdbcCommand cmd = new OdbcCommand(sql, con);
        da.SelectCommand = cmd;
        da.Fill(ds);
        try {
                          rt = ds.Tables[0]; }
        catch {
           rt = null;
        }
        return rt;
     }
</script>
```

JAVA

COME APRIRE CHIUDERE VANO CD DA JAVA

Java non offre nativamente il supporto per l'accesso a tale funzionalità. Possiamo ovviare a questa limitazione appoggiandoci al codice messo a disposizione dal nostro sistema operativo, utilizzando JNI. Con questa classe possiamo usare delle semplici istruzioni come queste:

```
code:
  String drive = "E:";
  if(!DiscTrayFacility.open(drive))
  {
    System.err.println(DiscTrayFacility.getErrorMessage());
  } else
  {
    System.out.println("Lettore " + drive + " aperto");
  }
```

IL TIP DEL MESE

COME APRIRE IL VANO DEL CD IN LINUX CON C++

Tip fornito da Domenico Testa

```
#include <iostream>
#include <string>
using namespace std;
#include <errno.h>
#include <fcntl.h>
#include <sys/ioctl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <linux/cdrom.h>
int main(int argc, char *argv[])
{
   string device = "/dev/cdrom";
   if(argc > 1)
  {device = argv[1]; }
   // Apertura del device:
   int drive = open(device.c_str(), O_RDONLY | O_NONBLOCK);
   if(-1 == drive)
   {
      cerr << "Impossibile aprire il device " << device
      << ".Codice errore " << errno << endl;
    return 1;
  }
  // Apertura del cassettino:
  if(ioctl(drive, CDROMEJECT))
  {
    cerr << "Impossibile aprire il vano cd del device " << device
    << ". Codice errore " << errno << endl;
  }
  cout << "Vano cd aperto per " << device << endl;
  // Chiusura del cassettino:
  if(ioctl(drive, CDROMCLOSETRAY))
  {
    cerr << "Impossibile chiudere il vano cd del device " << device
    << ". Codice errore " << errno << endl;
  }
  cout << "Vano cd chiuso per " << device << endl;
  return 0;
  }
```

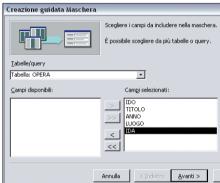
Data entry avanzati in Access

Il data entry è un'operazione effettuata spesso da utenti che non hanno esperienza approfondita in campo informatico. È importante realizzare maschere di input che facilitino il compito a chi deve inserire i dati. Bisogna progettare prima e sviluppare poi una form che sia in grado di accogliere nuovi record per una generica tabella di Access. La procedura è banale se si tratta di una tabella che non ha alcuna relazione con altre tabelle. Invece, nel caso in cui vi siano associazioni, ad esempio del tipo uno a molti organizzare un input efficiente richiede maggiore accortezza. Scartando a priori la possibilità di inserire nella tabella la chiave esterna, ovvero l'identificativo della tabella relazionata; bisogna trovare un modo per inserire nella form di input il riferimento testuale al record, che non sia un asettico numero, come la chiave esterna. Nell'esempio considereremo una galleria di opere d'arte, dove le due tabelle OPERA e ARTISTA contengono rispettivamente informazioni sulle opere d'arte –quadri- e sugli artisti che le hanno dipinte. Nella tabella opera tra i vari attributi appare IDA come riferimento all'artista, precisamente alla sua chiave. È questo l'attributo che crea un legame tra le due tabelle. Nella maschera di input vogliamo leggere un record completo della tabella opera, ma allo stesso tempo quando si tratterà di specificare l'artista che ha prodotto l'opera vogliamo far apparire da un menu di nomi degli artisti contenuti nella tabella apposita. In automatico verrà associata al nome dell'artista la sua chiave.

Fabio Grimaldi

‹1› CREAZIONE GUIDATA DI UNA MASCHERA

Il primo passo, dopo aver lanciato Access, è quello di selezionare dal raccoglitore degli strumenti la maschera. Decidiamo, per semplificare la procedura, di avviare la creazione guidata. Ricordo che l'obiettivo è quello di produrre una form che realizzi l'input per la tabella *OPERA*. Il DB contiene già dei dati per effettuare la prova.

‹2› AVVIO DELLA PROCEDURA GUIDATA

Nella finestra di dialogo selezionare la tabella *OPERA*, per la quale si vuole realizzare il data entry. Successivamente, cliccando sulla doppia freccia si decide di riportare nella maschera tutti gli attributi della tabella. Al termine della procedura, dopo aver scelto il tipo di maschera e il layout, si opterà per visionare la modalità struttura.

‹3› AGGIUNTA DI UNA CASELLA COMBINATA

Dal raccoglitore di strumenti si aggiunge una casella combinata per creare il menu. La si aggiunge alla maschera che è ancora in formato struttura. Si rimuove l'attributo IDA, ora sostituito con il nuovo oggetto e si sistema la casella combinata e il titolo. Cliccando su di essa si avvia la procedura di associazione di valori alla casella.

‹4› RICERCA DEI VALORI SULLA TABELLA

Si opta per la ricerca di valori sulla tabella. La finestra seguente ci permette di selezionare la tabella *ARTISTA* e il campo *NOMIN*. Adesso il menu apparirà in modo corretto. Resta solo da associare ai valori che appariranno nella lista la corrispondente chiave. Di fatto noi sceglieremo una nominativo e invieremo un codice.

‹5› MEMORIZZAZIONE IN UN ATTRIBUTO CHIAVE ESTERNA

La procedura in corso chiede se memorizzare il valore selezionato o altro. Indichiamo la seconda opzione e così scegliendo tra gli attributi preleviamo *IDA*. Così *NOMIN* di *ARTISTA* sarà associato alla sua chiave *IDA* che è anche chiave esterna della tabella di cui stiamo producendo la maschera di input.

‹6› OSSERVIAMO IL RISULTATO FINALE

Non resta che apprezzare il risultato ottenuto. Chiudendo e salvando la maschera si ha la possibilità di riaprirla in modalità di esecuzione.
Si vede come per una determinata opera che si inserisce, si possa scegliere tra una lista di artisti. Vale anche per la visualizzazione.
Se l'artista non è presente nella lista bisogna inserirlo preventivamente nella tabella *ARTISTA*.

La prima applicazione visuale in DEV-C++

In passato per gli sviluppatori C e C++ il passaggio dalla programmazione tradizionale alla programmazione visuale ha segnato un elemento di difficoltà. Produrre codice utilizzando le win API, non sempre è stata una passeggiata. Il desiderio di molti programmatori è scrivere il codice puro che risolve il problema che sono chiamati a risolvere, senza doversi occupare dell'interfaccia dell'applicazione. DEV-C++, fornisce alcune facilities interessanti che consentono al programmatore di svincolarsi dalla realizzazione dell'interfaccia grafica. In effetti, si riesce quasi totalmente a separare il codice per la soluzione del problema con quello per la gestione delle risorse grafiche. Ma come gestire queste ultime? La teoria a tale proposito consta di una buona quantità di nozioni. Un approccio tipico dei programmatori, tenta di abbattere le eventuali carenze teoriche facendo uso di esempi. In Dev-C++ vi sono un sufficiente numero di progetti che aiutano il programmatore sprovveduto a riguardo. Osserviamo uno degli esempi per acquisire l'abilità di manipolare finestre. Noteremo che facilmente andando a spulciare tra le linee di codice dell'esempio si comprenderanno alcune nozioni per la gestione delle winAPI. L'esempio genera semplicemente una finestra con una sola scritta, la classica "Hello world!" è inserisce un bottone per la chiusura della stessa. Vediamo come procedere per ottenere tale risultato.

Stefano Vena

‹1› APERTURA DEL PROGETTO DI ESEMPIO

Dopo aver lanciato l'applicazione DEV-C++, dall'ambiente di sviluppo, apriamo un nuovo progetto. Dal menu file selezioniamo la voce apri progetto e andando a seguire il percorso C:\dev-cpp\examples\wintest selezioniamo appunto il progetto wintest, ossia l'insieme di file che implementano l'applicazione descritta.

‹2› COMPILAZIONE DELL'INTERO PROGETTO

Per produrre l'output è necessario compilare l'intero progetto. Ovviamente, Dev-C++ cercherà tra i file presenti nel progetto una radice; per così dire. Un file che richiama gli altri. Nel caso specifico il progetto è semplice e contiene un solo file che è main.c, codificato per semplicità in C. Si, perché ricordo, che Dev-C++ compila C++ e C.

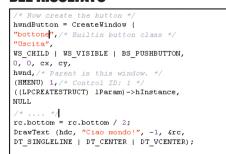
‹3› RISULTATO DELLA COMPILAZIONE

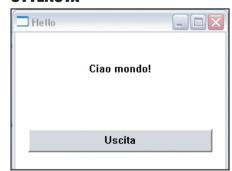
Cliccando sul tasto di compilazione si potrà osservare sulla finestra di stato che appare, la progressione dell'intera compilazione, l'eventuale link e la produzione del file eseguibile. Se si riscontrano errori questi verranno opportunamente segnalati. Nel nostro caso come mostrato, tutto è andato per il meglio e quindi possiamo procedere nell'osservazione del risultato.

‹4› L'APPLICAZIONE VIENE ESEGUITA

Dalla barra degli strumenti, il bottone accanto alla compilazione ci consente di eseguire il progetto, ossia il file eseguibile prodotto. Osserviamo il semplice quanto funzionale risultato, una finestra con la frase "Hello world!" al centro e un bottone per chiudere la stessa. Premiamo questo ultimo. Ci apprestiamo a personalizzare il risultato, un modo per comprendere gli elementi grafico visuali.

‹5› PERSONALIZZAZIONE DEL RISULTATO

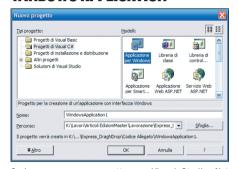
```
/* Now create the button */
hwndButton = CreateWindow (
"bottone",/* Builtin button class */
"Uscita",
WS_CHILD | WS_VISIBLE | BS_PUSHBUTTON,
0, 0, cx, cy,
hwnd,/* Parent is this window. */
(HMENU) 1,/* Control ID: 1 */
((LPCREATESTRUCT) lParam)->hInstance,
NULL
/* .... */
rc.bottom = rc.bottom / 2;
DrawText (hdc, "Ciao mondo!", -1, &rc,
DT_SINGLELINE | DT_CENTER | DT_VCENTER);
```

Andando a manipolare il file main.c si possono comprendere gli elementi base utilizzati. Per farlo modifichiamo alcuni di essi, come le scritte che appaiono all'interno della finestra e sopra il bottone. Italianizziamo l'applicazione. Il percorso da seguire per la comprensione è quindi tracciato, modificare alcuni elementi per testare gli effetti di cambiamento sull'eseguibile.

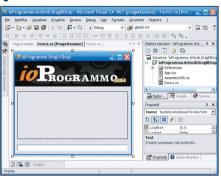
‹6› NUOVA APPLICAZIONE OTTENUTA

Si ottiene una nuova applicazione, praticamente identica alla precedente ma che ha le scritte in italiano. Notiamo anche che si tratta di una finestra standard che può essere ridimensionata, spostata, minimizzata, ingrandita e chiusa. Per terminare scegliamo di cliccare sul tasto Uscita. Pochi passi è abbiamo raggiunto l'obiettivo.

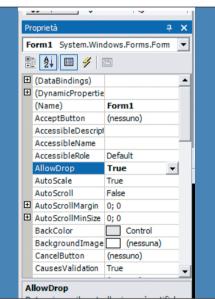
Abilitiamo il trascinamento nelle nostre applicazioni .Net

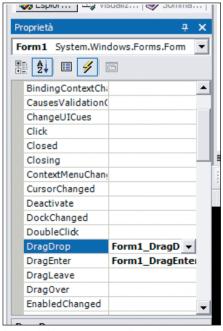
In molti programmi siamo ormai abituati all'uso dell'operazione comunemente chiamata Drag and Drop. Si tratta di una operazione che ci consente di trascinare un elemento (una immagine, un testo, un link) all'interno di un altro software e di elaborarlo. Pensiamo ad esempio ad un comune client di posta: quando vogliamo inviare un file in allegato ad una mail, la cosa più comoda da fare è quella di trascinare l'allegato direttamente nel messaggio. In questo breve articolo vedremo come sia semplice attivare la stessa funzionalità nei nostri programmi. Realizzeremo una semplice applicazione in cui potremmo trascinare un file e vederne l'anteprima. Sarà poi semplice implementare eventuali operazioni da compiere con il file trascinato. Lo strumento che utilizzeremo sarà VisualStudio.net, il linguaggio che abbiamo scelto è l'ormai noto C#

Michele Locuratolo

‹1› UN NUOVO PROGETTO WINDOWS APPLICATION

Creiamo un nuovo progetto con Visual Studio .Net 2003 scegliendo come tipo di applicazione "Applicazione per Windows". Scegliamo il nome e la posizione e clicchiamo su OK.

‹2› AGGIUNGIAMO QUALCHE ELEMENTO

Costruiamo la nostra semplice applicazione aggiungendo al form vuoto una PictureBox ed una TextBox. La TextBox visualizzerà il percorso del file trascinato mentre la PictureBox mostrerà una anteprima del file.

‹3› SETTIAMO LE PROPRIETÀ

La cosa importante da fare è specificare che il form deve essere in grado di gestire il trascinamento di un file. Per farlo, selezioniamo il form e nelle proprietà settiamo a True la voce "AllowDrop".

‹4› GESTIAMO GLI EVENTI

Ora che il form può accettare il trascinamento di un file, dobbiamo gestire tale evento! Attiviamo quindi la scheda eventi del form e abilitiamo DragDrop e DragEner facendo doppio click su ognuno di essi.

‹5› LAVORIAMO CON I FILE

```
private void Form1_DragDrop(object sender,
System.Windows.Forms.DragEventArgs e) {
  string filePath = string.Empty;
  foreach (string fileName in (string[])e.Data.GetData(
                                    DataFormats.FileDrop))
    filePath = fileName;
tbxPath.Text = filePath;
  string ext = tbxPath.Text.Substring(
                              filePath.IndexOf("."));
  if ( ext == ".gif" | ext == ".jpg" )
  {
    pictureBox1.Image = new Bitmap(tbxPath.Text);
  }
  else {
    MessageBox.Show("...");
  }
}
```

Nell'evento DragDrop recuperiamo il nome ed il percorso del file trascinato con string fileName in (string[])e.Data.GetData(DataFormats.FileDrop) e inseriamolo nella textbox. Se l'estensione del file è di .gif o .jpg, la inseriamo nella PictureBox per visualizzarne una anteprima.

‹6› L'APPLICAZIONE FINALE

Ecco in nostro piccolo programma in funzione. Con lo stesso principio sarà possibile salvare l'immagine trascinata, copiare del testo, inserire un dato in un Data Base.

Scrivere e leggere un piccolo file di configurazione

Agni buon programma che si rispetti ha delle impostazioni da salvare per la sua configurazione. Se ci troviamo nella situazione di dover salvare i dati del nostro programma abbiamo due alternative usare il file di registro di windows oppure affidarci ad un tradizionale file di configurazione. Questo genere di file a differenza del file di registro ci permette di mantenere le impostazioni del programma anche dopo una formattazione (a patto che esso sia stato salvato). Quindi ora andiamo a vedere come si realizza un file di configurazione e come leggere le informazioni salvate su esso. Ciò di cui abbiamo bisogno è un compilatore c++, un editor di testo e un po' di dimestichezza con c++. Per realizzare il seguente esempio ho utilizzato Dev-C++ un ambiente di sviluppo c++ freeware molto leggero. Dev-C++ è scaricabile gratuitamente dal sito web: *www.bloodshed.net*, oppure potete usare la versione contenuta nel CD allegato.

Stefano Vena

‹1› CREARE UN NUOVO PROGETTO CONSOLE

Per prima cosa lanciamo dev-c++, poi dal sottomenu Nuovo del menu File scegliamo la voce Progetto, poi scegliamo dalla sezione basic il tipo di progetto "Console Application" diamo un nome al nostro lavoro e diamo l'ok.

‹2› LE PRIME RIGHE DI CODICE

```
#include <iostream>
#include <stdlib.h>
#include <fstream>

using namespace std;

int main ( int argc, char *argv[] ) {
```

Il progetto contiene già un file con un semplice un programma c++, frutto dei modelli predefiniti utilizzati dall'ottimo dev-C++.
Precisamente sono presenti due inclusioni *<iostream>* e *<stdlib.h>* (oppure *<cstdlib>* se usiamo *gcc3.4.x*) ed una semplice implementazione di main.
Aggiungiamo l'header *<fstream>* e andiamo al passo successivo.

‹3› SCRITTURA DEL FILE DI CONFIGURAZIONE

```
ofstream outcfg( "config.cfg" );

if( outcfg.fail() != 0 ) return -1;

outcfg <<   "Nome= " << "Mario" << endl;
outcfg << "Cognome= " << "Rossi" << endl;
outcfg <<   "Anni= " << 35    << endl;

outcfg.close();
```

La prima cosa da fare è dichiarare la variabile di tipo (ofstream) per lo stream di scrittura. Utilizziamo il costruttore che prende come parametro il nome del file così possiamo automaticamente creare il file di configurazione.
Se il file scelto esiste già esso verrà sovrascritto. Il passo successivo è controllare se il file scelto è stato creato correttamente. Per fare ciò controlliamo lo stream tramite il metodo "fail()". Il precedente metodo restituisce un valore uguale a zero se tutto è andato bene. Per scrivere nello stream utilizziamo l'operatore di inserimento (<<) proprio come facciamo quando scriviamo con cout sulla console.
La sintassi è semplice: inseriamo prima una breve descrizione poi uno spazio come separatore , poi il valore vero e proprio e in fine andiamo a capo tramite endl. Una volta scritti tutti i valori chiudiamo lo stream per applicare i cambiamenti.

‹4› LA LETTURA DELLA CONFIGURAZIONE

```
string temp;
string nome, cognome;
int anni;
ifstream incfg( "config.cfg" );
if( incfg.fail() != 0 ) return -1;
incfg >> temp >> nome;
incfg >> temp >> cognome;
incfg >> temp >> anni;
incfg.close();
```

Istanziamo quattro variabili da utilizzare come supporto durante la lettura. Creiamo subito dopo uno stream di lettura (ifstream). La variabile incfg viene inizializzata utilizzando il costruttore della classe ifstream che prende come parametro il nome del file da aprire. Nel processo di lettura utilizziamo l'operatore di estrazione (>>) a cascata. Estraiamo la breve descrizione dallo stream e la assegnamo alla variabile temp poi andiamo a riempire le altre variabili con i dati letti sul file. Anche in questo caso andiamo a chiudere lo stream.

‹5› STAMPA DELLE OCCORRENZE

```
cout << "Nome :";
cout << nome << endl;

cout << "Cognome :";
cout << cognome << endl;

cout << "Anni :"
cout  << anni << endl;

system("PAUSE");
```

Dopo aver letto tutte le occorrenze dal file andiamo a stamparle sullo schermo della console. Dopo aver stampato il risultato attendiamo la pressione di un tasto da parte dell'utente. Il tutto si risolve con poche righe di codice.
Utilizziamo l'istruzione sistem ("PAUSE") per mettere il programma in attesa fino alla pressione di un tasto. Le righe precedenti sono sufficientemente semplici da non richiedere commento.

‹6› IL RISULTATO

Se tutto è andato per il verso giusto questo è il risultato della lettura stampato sulla console. Graficamente non entusiasmante ma efficiente.

Nascondere la traybar

A volte potremmo sentire la necessità di dovere nascondere la traybar.
È il caso tipico di applicazioni che devono essere residenti nel sistema e continuare a svolgere il loro compito senza ingombrare il desktop dell'utente.
Ad esempio potremmo decidere di nasconderla per impedire all'utente di accedere al menu avvio oppure alla lista dei programmi aperti. Potrebbe anche accadere di voler nascondere la Tray Bar per sostituirla con una diversa.
Nell'esempio riportato di seguito andremo a vedere come ottenere nascondere o rendere visibile la tray bar in modo molto semplice utilizzando soltando le API di Windows e un po' di C++; All'inizio del nostro programma andremo a verificare se la tray bar è visibile, se lo è la nascondiamo altrimenti la rendiamo visibile, il tutto è veramente semplice. Ciò di cui abbiamo bisogno è un compilatore c++, un editor di testo e un po' di dimestichezza con c++. Per realizzare il seguente esempio abbiamo utilizzato il classico Dev-C++

Stefano Vena

‹1› CREARE UN NUOVO PROGETTO CONSOLE

Per prima cosa lanciamo *dev-c++*, poi dal sottomenu *Nuovo* del menu *File* scegliamo la voce *Progetto*, poi scegliamo dalla sezione *basic* il tipo di progetto *"Console Application"* diamo un nome al nostro lavoro e diamo l'*ok*.

‹2› LE PRIME RIGHE DI CODICE

```
#include <iostream>
#include <stdlib.h>
#include <windows.h>

using namespace std;

int main ( int argc, char *argv[] ) {
```

Il progetto contiene già un file con un semplice un programma c++, frutto degli schemi proposti dall'ambiente di sviluppo. Precisamente sono presenti due inclusioni *<iostream>* e *<stdlib.h>* ed una semplice implemenzione di main. Aggiungiamo l'header *<window.h>* e questo è sufficiente per andare al passo successivo.

‹3› DICHIARAZIONE DELLE VARIABILI

```
BOOL isVisible = FALSE;
HWND hWnd      = NULL;

hWnd = FindWindow("Shell_traywnd", "");

isVisible = IsWindowVisible(hWnd);
```

Le variabili necessarie sono due: una variabile di tipo *HWND* che riceve il valore dello handle associato alla tray bar ed un valore booleano, utilizzato come flag e indica se la barra è attualmente visibile oppure se è nascosta.
Una volta dichiarate, le due variabili vengono settate. Per prima cosa otteniamo l'handle della barra attraverso la funzione *FindWindow* e lo assegniamo alla variabile *hWnd* poi verifichiamo lo stato della barra *(visibile/nascosto)* attraverso la funzione *IsWindowVisible.*
Una volta ottenuta questa informazione siamo pronti per lo step successivo.

‹4› NASCONDIAMO LA BARRA

```
if( isVisible )
{
 SetWindowPos(hWnd, 0, 0, 0, 0, 0,
                SWP_HIDEWINDOW);

 cout << "Nascondo la TrayBar...\n";
}
```

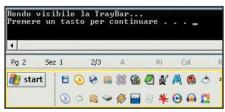
Prima d'ogni cosa andiamo a controllare lo stato della variabile *isVisible*, se essa ha come valore true allora la barra è visibile quindi procediamo a nasconderla tramite la chiamata a *SetWindowPos.* I parametri di tale funzione vanno impostati tutti a zero escluso il primo, il quale punta allo handle ottenuto in precedenza, ed il sesto che corrisponde al flag necessario per nascondere la barra. A questo punto la *tray bar* viene nascosta e non è più accessibile dall'utente. Le istruzioni per realizzare quanto proposto si riducono a pochissime righe di codice e non nascondono elementi di particolari complessità. L'unica accortezza è controllare i parametri di SetWindowsPos

‹5› VISUALIZZIAMO LA BARRA

```
else
{
 SetWindowPos(hWnd, 0, 0, 0, 0, 0,
      SWP_SHOWWINDOW);

 cout << "Rendo visibile la TrayBar...\n";
}
```

Se la variabile *isVisible* ha come valore *false* allora entriamo nella sequenza di funzioni di *"else"*. Ciò vuol dire che la barra è nascosta quindi andiamo a renderla visibile.
Per fare ciò ci serviamo nuovamente della funzione *SetWindowPos* ma al posto del flag *SWP_HIDEWINDOW* andiamo ad utilizzare *SWP_SHOWWINDOW.* Ora la barra è nuovamente visibile! Queste poche righe di codice sono molto potenti ma pericolose se non si presta attenzione. Infatti potremmo inserire, in un nostro programma, il codice necessario a nascondere la barra ma non quello per renderla nuovamente visibile costringendo l'utente al riavvio del sistema. Un classico BUG da cattiva programmazione, difficile da individuare a meno che non sia previsto il controllo da parte del programma

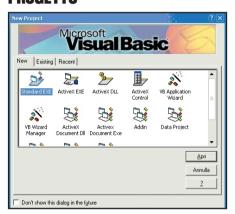
‹6› IL PRIMA E IL DOPO

Dopo la prima esecuzione del nostro codice la barra scompare.

Ora la barra è nascosta per visualizzarla nuovamente eseguiamo di nuovo il nostro codice e per magia la barra ritorna!

Realizzare un form circolare

Sei stanco dei soliti form quadrati? Allora questo è l'express che fa per te! Vedremo come è semplice ritagliare un form in Visual Basic 6 utilizzando solo due API di windows un vettore di punti ed un po' di fantasia. Benché questo esempio sia riportato in visual basic esso può essere riadattato a qualsiasi linguaggio di programmazione che abbia la possibilità di eseguire codice legato a API di windows. Il codice gira sotto ogni piattaforma microsoft dal windows 95 in poi. Gli unici requisiti richiesti sono un po' di dimestichezza con l'ambiente VB6, il VB6 stesso e una discreta conoscenza delle API di sistema. Isoleremo una funzione ritaglia all'interno della quale definiremo per mezzo di una funzione la forma che desideriamo fare apparire sul video. Questa form verrà utilizzata come una sorta di maschera che coprendo una form classica simulerà una finestra di forma circolare. Ovviamente è sufficiente modificare questa funzione con i valori che riterrete più opportuni per ottenere effetti anche complessi. Tutto quello di cui avete bisogno è VB6

Stefano Vena

‹1› CREARE UN NUOVO PROGETTO

Per prima cosa lanciamo VB6 nel wizard scegliamo il template "Standard exe" e poi scegliamo ok

‹2› LE API DI SUPPORTO

```
Private Declare Function SetWindowRgn Lib "user32"
    (ByVal Hwnd As Long, ByVal hRgn As Long, ByVal
    bRedraw As Boolean) As Long
Private Declare Function CreatePolygonRgn Lib
    "gdi32" (lpPoint As POINTAPI, ByVal nCount As Long,
    ByVal nPolyFillMode As Long) As Long
Const WINDING = 2
Private Type POINTAPI
  x As Long
  y As Long
End Type
```

Facciamo doppio click sul form di default al centro dell'ambiente di sviluppo e accediamo al codice. Nella zona di intestazione del form (nelle prime righe in alto) andiamo ad inserire le dichiarazioni delle funzioni esterne importate dalle librerie del sistema. Ne nostro caso dichiareremo SetWindowRgn e CreatePolygonRgn inoltre vanno dichiarate una costante (WINDING) ed una struttura POINTAPI. Suddetta struttura verrà utilizzata per decidere le coordinate della circonferenza che ritaglierà il nostro form.

‹3› LA FUNZIONE RITAGLIA: LE VARIABILI

```
Private Function Ritaglia(ByVal lati As Integer)

 Dim pts() As POINTAPI

 Dim rgn As Long
 Dim i As Integer

 Dim centroX As Single
 Dim centroY As Single
```

Ora andiamo a dichiarare una funzione privata che utilizzeremo per ritagliare il form.
Dopo la segnatura della funzione dichiaramo le variabili che utilizzeremo, siamo pronti per andare al passo successivo.

‹4› LA FUNZIONE RITAGLIA: INIZIALIZZAZIONE

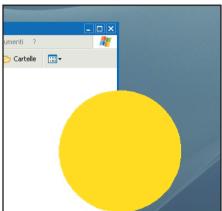
```
Me.ScaleMode = vbPixels
Me.BackColor = vbYellow
Me.Width = 300 * Screen.TwipsPerPixelX
Me.Height = 300 * Screen.TwipsPerPixelY
 centroX = Me.ScaleWidth / 2
centroY = Me.ScaleHeight / 2
ReDim pts(0 To lati)

For i = 0 To lati
       pts(i).x = centroX + 100 * _
          Cos(6.28 * i / lati)
    pts(i).y = centroY + 100 * _
          Sin(6.28 * i / lati)
Next i
```

Dopo aver dichiarato la variabile andiamo a preparare il form per essere ritagliato.
Innanzitutto andiamo a settare lo ScaleMode su vbPixels poi impostiamo il colore di sfondo sul giallo. A questo punto andiamo a ridimensionare il form (questa volta in twips) ricaviamo il centro del form e andiamo avanti.Gli ultimi passi da fare sono ridimensionare il vettore di POINTAPI e riempirlo con le coordinate di una circonferenza.

‹5› LA FUNZIONE RITAGLIA: IL RITAGLIO

```
  rgn = CreatePolygonRgn(pts(0), _lati + 1, _WINDING)
 Call SetWindowRgn(Me.Hwnd, rgn, True)
End Function
Private Sub Form_Load()
 Ritaglia 20
End Sub
Private Sub Form_Click()
 Unload Me
End Sub
```

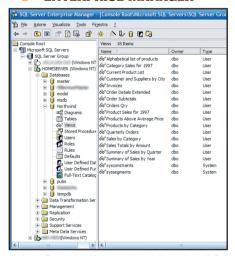
Tutte le funzioni dell'API di windows sono state esportate dal linguaggio C. La gestione dei vettori è differente rispetto a VB6. È possibile passare per riferimento il primo elemento di un vettore senza incorrere in errori. La variabile lati punta all'indice superiore dell'array pts, viene passato alla funzione incrementato di un'unità. Invocheremo la funzione Ritaglia all'interno dell'evento Form_Load.

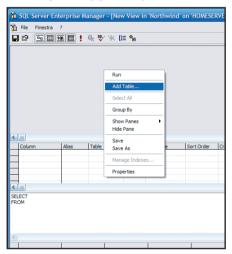
‹6› OSSERVIAMO L'APPLICAZIONE IN ESECUZIONE

Come avete potuto notare ritagliare un form è un'operazione abbastanza semplice ora partendo da questo esempio sbizzarritevi con le vostre forme. Tutto quello a cui dovete porre attenzione è la definizione della funzione di ritaglio. È anche importante osservare come nella definizione del passo 5 abbiamo utilizzato l'evento click del mouse per distruggere la form e liberare in questo modo le risorse di sistema. Altra nota importante è relativa al passaggio dei dati per riferimento invece che per valori

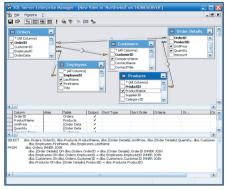
Semplifichiamo le query usando le Viste

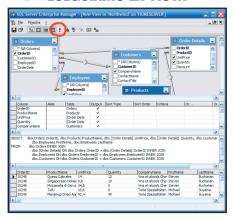
L'interrogazione di un Data Base è una operazione estremamente frequente nella maggior parte delle applicazioni. Concettualmente, tale operazione è semplice: usando una sintassi standard (SQL) si richiedono ad un Data Base dei dati che, sempre utilizzando la stessa sintassi, possono essere filtrati, ordinati, aggregati ecc. Quando abbiamo a che fare con Data Base complessi però, la stringa di richiesta dei dati può diventare complessa ed il suo utilizzo nel codice dell'applicazione può essere svantaggioso. Se utilizziamo Microsoft Sql Server come Data Base, abbiamo uno strumento molto potente per semplificare le interrogazioni: le viste. Una vista può essere intesa come una tabella virtuale che raccoglie dei dati e che può essere richiamata dalle nostre applicazioni come se fosse una tabella reale. Oltre a semplificare le interrogazioni, una vista ci permette in di astrarci dalla base dati in modo da non dover modificare la nostra applicazione se cambia la struttura del DB. Il che è un'ottimizzazione non da poco.

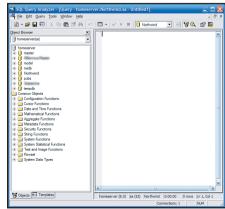
Michele Locuratolo

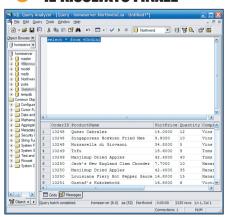
‹1› ENTERPRISE MANAGER

Apriamo Enterprise Manager (il tool di gestione di Sql Server), selezioniamo il Data Base di esempio Northwind e scegliamo il nodo Views

‹2› UNA NUOVA VISTA

Clicchiamo con il tasto destro del mouse su Views e scegliamo New View. Verrà aperto l'apposito editor in cui possiamo creare la nostra vista. Clicchiamo quindi con il tasto destro del mouse in un area vuota dell'editor e scegliamo la voce "Add Tables"

‹3› QUALI TABELLE?

Selezioniamo le tabelle di cui vogliamo recuperare i dati ed aggiungiamole all'editor con un doppio click. Una volta inserite le tabelle, selezioniamo i singoli campi che ci interesserà mostrare come risultato. Verrà creata una query uguale a quella che avremmo dovuto scrivere nel codice della nostra

‹4› ESEGUIAMO LA VISTA

Proviamo ora ad eseguire la nostra vista per verificare se il risultato è quello che ci aspettavamo. Per farlo clicchiamo sul punto esclamativo (!) rosso dell'editor ed analizziamo il risultato in basso. Se tutto è corretto, salviamo la Vista con il nome vOrdini e chiudiamo l'editor. Siamo pronti per il prossimo passo.

‹5› FACCIAMO UN TEST

Per testare la vista come se la stessimo eseguendo dalla nostra applicazione, avviamo il Query Analyzer. Un altro tool fornito insieme a Sql Server per inviare comandi al nostro Data Base.

‹6› IL RISULTATO FINALE

Recuperiamo i dati dalla nostra vista con la semplice stringa: "Select * from vOrdini" al posto di quella più complessa visibile in Fig. 3. Oltre a questo significativo vantaggio sintattico, se dovesse cambiare qualcosa nelle tabelle che compongono la vista, sarà sufficiente modificare essa e non la nostra applicazione. Il gioco è fatto!

SOFTWARE SUL CD

Apache 1.3.33/2.0.53

Il Web Server più usato al mondo

Le due versioni del Web Server progettato da Apache che da solo tiene in piedi una buona parte di Internet. Nonostante l'agguerrita concorrenza Apache rimane un web server incredibilmente usato. I suoi moduli sono praticamente illimitati, è estremamente leggero, viene utilizzato praticamente su tutte le piattaforme di Hosting del mondo. Se avete in mente di progettare un'applicazione Web non potete fare a meno di avere installato sulla vostra macchina in locale una versione di Apache. La 1.3.33 è stabilissima ed è consigliata in abbinamento ad esempio a PHP, la 2.0.53 pur essendo ormai consolidata, rimane una versione da usare con cautela.

Directory: /Apache

Dadabik 3.2

Un creatore di interfacce verso database

Si tratta di una Web Application scritta in PHP che consente di costruire altre Web Application basate su database.

Ad esempio se volete costruire un sito web che esponga un catalogo multimediale, non vi resta che informare DadaBik di quali campi si compone il catalogo in questione, di quali funzionalità volete che il vostro sito sia dotato, e lasciare a DadaBik il compito di generare la vostra interfaccia. Non è necessario avere competenze estese di programmazione, l'uso di DadaBik è piuttosto semplice

Directory: /DadaBik

wxWidgets 2.4.2

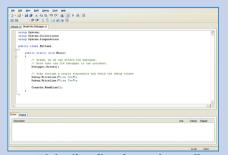
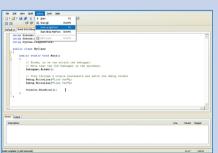
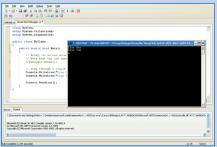
Comode librerie per lo sviluppo di interfacce grafiche

Interessantissime queste librerie, più volte le abbiamo utilizzate all'interno dei nostri programmi per realizzare degli esempi. Si tratta di librerie che consentono la creazione di interfacce grafiche, possono essere utilizzate da C++ ma anche da altri linguaggi come ad esempio Python. La cosa estremamente interessante è che consentono lo sviluppo di applicazioni completa-

Snippet Compiler 2

Per compilare un piccolo pezzo di codice

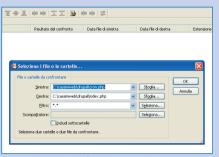
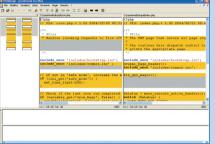
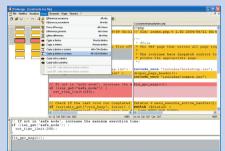
Un utility che consente di scrivere, compilare e far girare piccoli spezzoni di codice. Molto utile se si vuole provare il funzionamento di una porzione di codice senza per questo voler creare un completo progetto con Visual Studio prima di eseguire il tutto.

Directory /SnippetCompiler

1 Digita il codice che vuoi compilare all'interno della finestra principale dello snippet compiler. In ogni caso avrai a disposizione la sintax highlighting e la code complection

2 Quando sei pronto, avvia una sezione di debug utilizzando il menu Debug, oppure scegli un'opzione come "Build" per compilare il tuo codice velocemente

3 Per fare partire il programma, puoi utilizzare il classico tasto "start". La sezione di output verrà mostrata in basso nella finestra apposita come in figura

Winmerge 2 2.2

Per non trovarsi mai di fronte a versioni di codice differenti

Un tool che vi consente di paragonare due file e trovarne eventualmente le differenze. Utilissimo tutte quelle volte che avete apportato delle modifiche, ma non vi ricordate più qual è la versione più recente o esattamente cosa avete cambiato. Si tratta di una di quelle utility che non rivoluzionano la vita, ma che sicuramente contribuiscono a semplificare la routine giornaliera

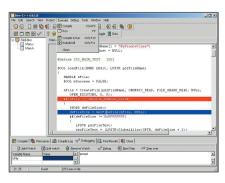
Directory /WinMerge

1 **Dal menu file, scegli apri. Apparirà una dialog box in cui inserire i dati relativi ai due file da comparare. Scegli il primo utilizzando il tasto sfoglia relativo a "sinistra" e il secondo utilizzando il tasto sfoglia relativo a "destra".**

2 **I due file verranno mostrati rispettivamente nei due riquadri proposti da winmerge. Le differenze nelle corrispettive righe saranno evidenziate con un colore giallo. Sarà possibile evidenziarle ulteriormente facendo doppio click sul testo da controllare.**

3 **Apporta tutte le modifiche che ritieni necessarie, se è il caso il menu unisci ti darà una mano. Quando hai finito puoi creare una patch automatica oppure salvare direttamente uno dei due file con le modifiche effettuate. Oppure tornare alla versione precedente.**

mente multipiattaforma, sono disponibili infatti sia in ambiente *nix che in ambiente Windows.
Continueremo sicuramente a parlarne in modo intensivo da queste pagine.

Directory: /WxWidgets

DevCPP 4.9.9.2

La versione più recente dell'ambiente più usato dai programmatori C++

Dev C++ è pratico, non ha costi, è molto funzionale leggere e versatile, questo lo rende un tool estremamente amato dai programmatori C++ e che si contrappone nella sua semplicità a strumenti ben più costosi.

È ovviamente dotato di tutte le funzionalità tipiche di un ambiente di programmazione come il sintax highliting e il code complexion, ma soprattutto è un'indispensabile se ritenete di volere o dover programmare in C++

Directory: /DevCPP

PHP 5.0.4

Uno dei linguaggi principe per il Web

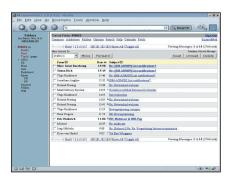
Se seguite ioProgrammo o più semplicemente siete dei programmatori Web, o ancora molto più semplicemente navigate su Internet, non potete non sapere che cosa è PHP. Si tratta del linguaggio con il quale sono sviluppate la maggior parte delle applicazioni internet esistenti.

Quasi tutto il software per il web si regge su PHP. La curva di apprendimento è bassissima, le funzionalità esposte elevatissime, certamente se avete intenzione di sviluppare per il web non potrete fare a meno di provare anche questo linguaggio come base per le vostre applicazione

Directory /PHP

SquirrelMail

La capostipite delle WebMail

Scritta interamente in Perl si tratta di un'applicazione che consente di leggere la posta direttamente da una comoda interfaccia Web. In realtà si tratta quasi della prima applicazione che è stata costruita con questo obiettivo, e se non si può dire che sia stato merito di SquirrelMail se si è diffusa su internet questa tendenza, possiamo affermare che nel dirlo non diciamo una cosa troppo lontana dalla verità.
SquirrelMail è dotata di un gran numero di Plugin e l'unico appunto

che gli si può muovere è quello di non essere efficace dal punto di vista delle possibilità di personalizzazione dell'interfaccia, inoltre non è multihoming, nonostante questo è tuttavia una scelta quasi obbligata per chi vuole esporre funzionalità di webmail sul proprio sito.
Directory /SquirrelMail

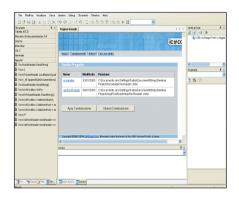
STRUTS 1.2.4

Il FrameWork per costruire applicazioni conformi all'MVC

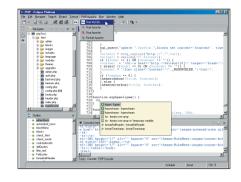
Di Struts ci siamo occupati più di una volta dalle pagine di ioProgrammo. Anche del patter Model View Controller abbiamo più volte discusso. Il pattern MVC stabilisce delle regoli tali che la logica di interfaccia, la logica di modello e quella di programmazione di un'applicazione siano completamente separate. Struts è un framework piuttosto solido che consente di creare applicazioni Web conformi allo standard MVC utilizzando Java e JSP
Directory /Struts

SharpDevelop 1.0.3

L'ambiente alternativo per la programmazione C#

Per chi non vuole affrontare i costi di Visual Studio.NET, per chi comunque vuole disporre di un ambiente abbastanza leggero per potere programmare le proprie applicazioni utilizzando C#, SharpDevelop sembra essere l'unica alternativa.

Da poco è possibile utilizzarlo anche per programmare in VB.NET, tuttavia la sua caratterizzazione è rivolta prevalentemente a C#. Si tratta di un prodotto ormai consolidato, rapido, visuale, economico e semplice da usare, alternativo a Visual Studio
Directory /SharpDevelop

PHPEclipse 1.1.3

Un plugin per usare Eclipse come ambiente di programmazione per PHP

Molti di voi conosceranno Eclipse, si tratta dell'ambiente "tuttofare" con cui è ormai possibile programmare qualunque tipo di linguaggio o quasi. Pur essendo fortemente orientato verso Java, Eclipse è estendibile tramite Plugin per essere utilizzato per diversi scopi. Il plugin che qui vi presentiamo estende Eclipse al fine di potere utilizzarlo per programmare applicazioni PHP
Directory /PHPEclipse

Eclipse 3.0.2

La nuovissima versione dell'ambiente di programmazione più innovativo del momento

Eclipse rappresenta uno dei maggiori punti di rottura con la concezione tradizionale di "IDE". Mentre fino ad ora la maggior parte degli IDE concentravano le proprie funzionalità per essere un supporto a un unico linguaggio, eclipse si configura come un ambiente aperto estendibile tramite plugin.

Questo garantisce che comportamenti simili attribuibili all'ambiente siano condivisi da più linguaggi di programmazione. In questo modo si abbraccia l'idea che il moderno programmatore debba essere in grado comunque di conoscere più di una tecnica e saperla utilizzare al momento giusto. Eclipse è il tool ottimale per supportare questa idea.
Directory /Eclipse

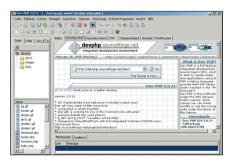
PircBot 1.4.4

Un bot irc programmabile in Java

Interessante questa idea di fornire agli amanti di IRC una serie di API che gli consentono di programmare un bot utilizzando JAVA.
Normalmente chi si dedica a questo tipo di applicazioni utilizza Eggdrop e programma in TCL, in tutti e due i casi non si tratta di tecniche di larga diffusione, viceversa Java è un linguaggio che gode di un'immensa popolarità, per cui il poter programmare dei Bot attraverso la flessibilità di java rappresenta senza dubbio un'opportunità interessante
Directory /PircBot

DevPHP

Un IDE leggerissimo per programmare PHP

Scritto sotto licenza GPL e perciò decisamente gratuito, DEVPhp è uno dei tanti fiori all'occhiello del mondo OpenSource.

Leggerissimo, ma completo, dotato di tutte le funzionalità tipiche di un IDE di alto livello, si pone come una scelta interessantissima per chi vuole programmare in PHP avendo a disposizione il comodo supporto del Syntax Highlighting, del code complexion, della gestione del file system e così via
Directory /DevPHP

Tomcat 5.5.9

L'application Server per JSP

Se siete dei programmatori JSP o aspirate a diventarlo, il vostro appli-

cation Server di riferimento è sicuramente Tomcat.
Giunto alla sua versione 5.5.9 è ormai da considerarsi decisamente un punto di riferimento per quanti utilizzano queste tecnologie per lo sviluppo di applicazioni Web. Non è certo un mostro di leggerezza, o un esempio di semplicità d'utilizzo, tuttavia è sicuramente una strada obbligata per programmare in JSP

Directory /Tomcat

Thinlet

Le API per creare interfacce usando XML

Ne parliamo diffusamente in questo stesso numero di ioProgrammo, con un bell'articolo di Daniele de Michelis. Le Thinlet sono un set di API che consente di definire in comodi file XML delle interfacce grafiche per applicazioni Java. È altrettanto semplice richiamare i file in questione dal codice Java, programmare gli eventi che corrispondono ad azioni compiute su bottoni e pannelli e tenere in questo modo separate la logica di programmazione da quella di definizione dell'interfaccia. L'utilizzo di XML per la definizione dell'aspetto dell'applicazione è decisamente interessante, se si pensa alla comodità di dover agire su semplici file di testo.

Directory /Thinlet

Hibernate 3.0

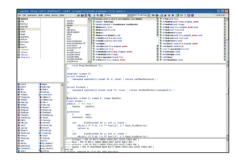
Il tool per la persistenza dei dati in JAVA

I programmatori sono abituati a pensare in termini di classi ed oggetti. Non c'è programma che prescinda ormai dalla logica della programmazione OOP. Tuttavia i database SQL ragionano ancora in termini di puro linguaggio, non c'è nessuna correlazione fra un dato e l'oggetto che lo rappresenta. Hibernate aggira questo ostacolo, ponendosi come framework che maschera un database relazionale con una logica OOP.
In questo modo i programmatori Java possono ad esempio accedere a un qualunque database pensando alle righe e alle colonne che lo rappresentano in termini di oggetti e non di entità relazionali.

Directory: /Hibernate

Ultimate ++

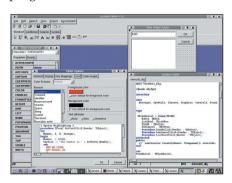
L'IDE più innovativo per C++

Se DEV C++ rappresenta ormai uno standard per i programmatori C++, Ultimate ++ si sta dimostrando un'ottima alternativa. Leggero, veloce, dotato di alcune estensioni che lo rendono in parte RAD, viene distribuito con il compilatore MinGW.

È dotato naturalmente di tutte le caratteristiche di un buon IDE, tuttavia dispone di un'organizzazione che lascia rapidamente intendere che da questo ambiente dovremo attenderci delle grosse novità nel futuro, sia in termini di prestazioni che di completezza

Directory /Ultimatepp

Lazarus 0.9.6

Il clone FreeWare di Delphi

Delphi è stato il capostipite dei moderni ambienti RAD e forse quello che meglio di ogni altro ha interpretato e continua a interpretare il senso della programmazione RAD. Il linguaggio che sta al di sotto di Delphi è l'Object Pascal che purtroppo però non ha mai incontrato il favore dei programmatori.

Le ragioni di questa poco affetto nei confronti dell'Object Pascal sono storiche prima e di marketink commerciale dopo, tuttavia Delphi è e rimane un tool RAD estremamente sofisticato adatto alla creazione di applicazioni di ampio respiro ed estremamente professionali. Lazarus è il clone FREE di Delphi. Se pur non è dotato di tutte i raffinati meccanismi che fanno del Delphi attuale lo straordinario strumento che è, lazarus è comunque un tool efficacissimo che da un lato interpreta la logica della programmazione RAD ai massimi livelli, dall'altro consente di programmare a oggetti utilizzando Object Pascal che a sua volta è e rimane uno dei migliori linguaggi per la OOP.

Directory /lazarus

Drupal 4.6.0

Il principe delle piattaforme di Blogging

I Blog sono stati il fenomeno dell'ultimo anno. Non c'è persona che abbia un minimo di feeling con internet che non disponga del proprio Blog dove esporre i propri interessi o semplicemente annotare qualche pensiero. Drupal è attualmente una delle piattaforme per la costruzione di Blog maggiormente diffusa. Da notare che abbiamo usato il termine: "piattaforma" di fatto Drupal consente di riunire sotto un'unica applicazione molti blog.

Ogni utente registrato di un sistema Drupal diventa infatti un possessore di Blog. Dispone di strumenti tali da poter pubblicare rapidamente contenuti personalizzati. Questa nuova versione è finalmente compatibile con PHP 5.0, sono state infine migliorate le funzioni che consentono di personalizzare maggiormente i singoli Blog. Rimangono invariate le già ottime funzioni per la creazione di moduli personalizzati. Nel complesso un buon tool.

Directory /Drupal

PHP Java Bridge
Il ponte fra Java e PHP

Ne parliamo in questo stesso numero di ioProgrammo. Si tratta del software che consente di utilizzare classi Java direttamente da PHP. In questo modo è abbastanza semplice per un programmatore Java costruire le proprie classi e poi riutilizzarle anche sul web. Ad esempio una classe per il calcolo dei numeri primi compresi in un certo range, deve essere semplicemente richiamata da PHP per poter funzionare, ma analogamente funzionerà senza problemi anche in un ambiente standalone. Si tratta di un tool da provare sia per i programmatori PHP che per quelli Java, non mancate di leggere l'articolo qui proposto

Directory /PHPJavaBridge

Python 2.4.1.
Il linguaggio emergente

Un linguaggio di scripting, multipiattaforma, orientato agli oggetti. Tre caratteristiche che rendono questo linguaggio piuttosto appetibile. Non per niente tutte le classifiche mondiali sulla diffusione dei linguaggi di programmazione lo individuano come quello maggiormente in ascesa. Python è elegante, estensibile, con una curva di apprendimento non elevatissima. Viene utilizzato per programmare moltissimi tool di gestione dei sistemi Unix, ma sta trovando una rapida applicazione anche sui sistemi windows e nello sviluppo di applicazioni crossplatform.

Directory /Python

Mysql 4.1.11 – Beta 5.0.3
Uno dei DB server più usati al mondo

Se Apache e PHP fanno da supporto alla maggior parte delle applicazioni che girano oggi su Internet, è anche vero che quasi tutte queste applicazioni sfruttano almeno in parte un database MySQL come sostegno per conservare i dati che manipolano. MySQL è estremamente leggero, veloce, flessibile, inoltre è multipiattaforma. Offre caratteristiche di tutto rispetto, che vanno dal supporto alle transazione alla ricerca full text, è perfettamente integrato con PHP. Tutte queste ragioni ne hanno fatto un leader per quanto riguarda il settore dei database. In questo numero oltre alla versione 4.1.11 ormai consolidata vi presentiamo anche la nuovissima Beta 5.0.3 da non utilizzare in ambienti di produzione, ma piuttosto intrigante per capire da soli qual è la direzione che uno dei database più usati al mondo sta usando.

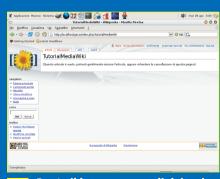
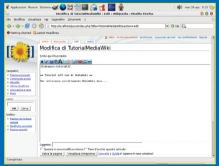
Directory /MySQL

Mono 1.6.0
Programmare in tecnologia .NET su tutte le piattaforme

Ce ne parla estensivamente Alessandro la Cava in questo stesso numero di ioProgrammo. Mono è la versione

MEDIAWIKI 1.4.0
L'applicazione preferita per la creazione di documenti

Riproponiamo MediaWiki ormai da qualche numero. Lo facciamo perché realmente crediamo che si tratti di uno di quei software utilissimi per un programmatore o per chi scrive la documentazione relativa a un software. Di fatto è ormai uno standard, si tratta per esempio dell'applicazione che fa da substrato a wikipedia. Ma in realtà trovate un wiki in quasi ogni sito che fa da supporto a un software. Per quelli che ancora non lo sapessero si tratta di un'applicazione che consente di scrivere documentazione in modo collaborativo e multimediale. Le pagine vengono create semplicemente puntando il browser su contenuti che non esistono fino a quel momento e i link vengono aggiornati in modo dinamico.

Directory /MediaWiki

1 Punta il browser su un link inesistente. Assicurati che l'url da te inventato abbia un titolo significativo. Ti apparirà una pagina che ti avverte che la pagina non esiste e ti chiede se vuoi crearla.

2 Scrivi il tuo articolo utilizzando la sintassi classica del Wiki. Puoi aiutarti con uno dei bottoni della maschera di edit. Nell'esempio abbiamo utilizzato il marcatore *{{category:tutoria}}* per indicare una categoria.

3 Clicca sul bottone salva la pagina. Il risultato sarà una pagina formattata secondo un layout corretto, inoltre a fondo pagina comparirà l'elenco delle categorie di appartenenza così come le abbiamo indicate nella maschera di edit.

OpenSource della piattaforma .NET. Ha dalla sua parte la possibilità di girare in multipiattaforma, per cui se scrivere un'applicazione C# compatibile Mono in ambiente Windows avrete la certezza che la stessa versione funzioni anche in ambiente Linux. Non è certo una caratteristica da poco se pensate alla possibilità di utilizzare le tecniche che meglio conoscete svincolando i clienti dalla necessità di dover investire verso un sistema piuttosto che un altro.
Directory /Mono

Irrlicht 0.9

L'attesissima nuova versione del motore 3D più in voga del momento

Di irrlicht abbiamo parlato tantissimo negli scorsi numeri di ioProgrammo. La serie condotta da Alfredo Marroccelli su queste stesse pagine è stata una delle più apprezzate degli ultimi numeri.

Si tratta di un engine 3D molto leggero e facile da usare, viene utilizzato da C++ è compatibile con OpenGL e DirectX ed è estremamente flessibile. Si tratta di un must per i programmatori di VideoGames o per chi aspira a diventarlo.
Directory /irrlicht

Icecast 2.2.0

Il server di streaming

Non è la prima volta che vi proponiamo questo server, tuttavia la grande diffusione che le radio online stanno attraversando ci convince che sia un tool da tenere in considerazione. Si tratta di un server che prende in input da un encoder dei suoni e li proietta in streaming su internet pronti per

essere ascoltati tramite ad esempio WinAmp o Windows Media Player.
È la base della costruzione di una radio online, o molto più semplicemente della diffusione della musica creata da voi su internet.
Usatelo ovviamente per fini legali.
Directory /Icecast

EasyPHP 1.8

Per installare un server PHP in modo semplice

Molti di voi avranno provato ad avvicinarsi a PHP. Il primo passo solitamente è creare un'installazione locale composta da Apache+PHP+MySQL, l'operazione non è complicata ma per chi è alle prime armi spesso si rivela un ostacolo insormontabile. EasyPHP è un tool che tramite pochi clic installerà per voi un sistema completo adatto ad essere utilizzato come server di prova per le vostre applicazioni PHP, svincolandovi così dalle operazioni di installazione, che si addicono più a un sistemista che ad un programmatore
Directory /EasyPHP

Jedit 4.2

L'editor leggero per programmatori Java

Realizzato completamente in Java, jEdit è un completo editor di testi per programmatori disponibile per numerose piattaforme tra cui MacOS X, OS/2, GNU/Linux (Unix) e Windows. Il programma, per l'immediatezza e la semplicità di utilizzo, è molto usato per scopi didattici e da programmatori non professionisti.

jEdit integra un completo linguaggio di macro e dispone di un'architettura facilmente estensibile mediante l'utilizzo di numerosi plugin facilmente gestibili tramite il "plugin manager". Inoltre, nonostante sia stato creato

SysLinux 3.0.7

Per creare periferiche di installazione di linux

Questo è realmente un software molto "stuzzicante" e che solletica la fantasia di ogni appassionato di informatica. Consente di creare dischetti di installazione di linux. Utilizzare syslinux è davvero molto semplice. E' sufficiente copiare i file del kernel su un dischetto è poi usare syslinux da riga di comando per ottenere un disco bootable con Linux dentro. Nello stesso pacchetto ci sono anche isolinux e extlinux per creare CDRom bootable contenenti linux e addirittura per creare una directory bootable in rete da cui far partire il sistema operativo sfruttando le caratteristiche dei moderni bios. Ora tutto questo è veramente interessante. Potete ad esempio creare la vostra distribuzione, sfruttando ad esempio BusyBox, o creare una penna USB bootable, o utilizzare questo sistema per mettere linux su delle memory card che poi svolgano dei compiti specifici.
Insomma le possibilità di applicazione sono veramente tante.
È sufficiente un po' di fantasia per realizzare cose davvero utili e divertenti.

Directory /SysLinux

per sviluppare prevalentemente applicazioni Java, dispone di funzionalità per indentare ed evidenziare il codice per oltre ottanta linguaggi di programmazione. Per poter installare e utilizzare jEdit è necessario disporre del J2SDK 1.3 o 1.4.
Directory /Jedit

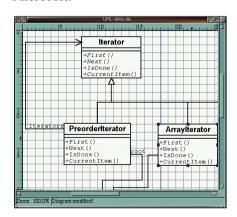
J2SE 1.5.0.2

Il framework essenziale per programmare in Java

Se siete dei programmatori java non potete prescindere dall'installare il JDK, si tratta del framework di base che contiene il compilatore, le utility e le librerie per potere iniziare a lavorare con Java. Quello che vi presentiamo è il secondo aggiornamento della versione 1.5 rilasciato da brevissimo tempo e contenente alcune migliorie dal punto di vista della security e moltissimi Bug Fix che migliorano in linea generale l'usabilità e l'affidabilità del linguaggio
Directory /J2SE

Dia 0.94

Crea diagrammi di flusso in modo semplice

Molto spesso nella programmazione di un software, o anche semplicemente nell'illustrare il ciclo di vita di un'applicazione o di una situazione si fa ricorso a diagrammi di flusso. Uno dei leader indiscussi del mercato in questo settore è senza dubbio Visio di Microsoft.

Dia non pretende di avere le stesse funzionalità di Visio e neanche la stessa complessità ma si pone come un software leggero diffuso sotto licenza OpenSource che può essere usato in modo conveniente in tutte quelle situazioni in cui è necessario generare un diagramma di flusso
Directory /Dia

James 2.2.0

Il server di posta di Apache

JAMES è l'acronimo anagrammato di Apache Java Enterprise Mailserver. Si tratta di un server SMTP, POP3 e NNTP interamente scritto in java. L'idea di base di Java è scrivere un mail server interamente portabile. L'utilizzo di Java in questo senso ha centrato completamente l'obiettivo. In questo numero di ioProgrammo abbiamo analizzato alcuni spezzoni del codice sorgente di James proprio per farci un'idea di come esso gestisca l'interazione con i server di Black List. Come si vede e come è classico dell'OpenSource l'estrema portabilità e la disponibilità del codice sorgente sono i punti di forza di progetti di questo genere.
Directory /James

J2ME Polish

Il costruttore di GUI per device mobile

J2ME Polish è forse un precursore dei tempi. Si tratta infatti di un software il cui scopo è supportare il programmatore nella creazione di interfacce destinate a dispositivi portatili quali ad esempio cellulari o palmari. Ovviamente la base su cui si fonda è J2ME ormai onnipresente in qualsiasi applicazione per dispositivi del genere, tuttavia invece la definizione dell'interfaccia basa le sue caratteristiche su semplici file HTML un po' come abbiamo visto fare a thinlet in questo stesso numero di ioProgrammo. Interessante è anche il fatto che J2MEPolish sia distribuito sotto licenza GPL.
Directory /J2MEPolish

Installer2Go

Un sistema facile e gratuito per realizzare pacchetti di installazione

Un tool per la creazione di file autoinstallanti che non impegna lo sviluppatore nel dover imparare l'ennesimo linguaggio di scripting: con una interfaccia che punta tutto sul frag&drop, realizzare pacchetti di installazione diventa un vero gioco da ragazzi. Benché gratuito, Installer2Go va incontro a tutte linee guida per la certificazione WindowsXP. Versione dimostrativa: alla fine di ogni istallazione si è rimandati ad una pagina pubblicitaria che fa riferimento al produttore di Installer2Go.
Directory /Installer2Go

Firebird 1.5

Un database evoluto e multipiattaforma

Il motore del database Firebird è stato creato da un team indipendente di sviluppatori volontari partendo dal codice sorgente del database InterBase prodotto da Borland (allora Inprise) e distribuito con licenza IBL (InterBase Public License) il 25 Luglio 2000. Firebird è un database relazionale (RDBMS) che supporta in modo quasi completo lo standard ANSI SQL-92, ed è disponibile per sistemi GNU/Linux, Windows e può essere eseguito su una varietà di piattaforme UNIX. Inoltre, Firebird offre prestazioni elevate e pieno supporto per stored procedures e triggers.
Directory /Firebird

XAMPP for Windows 1.4.13

ServerWeb, FTP, DBMS, PHP e Perl tutto in uno

Molti sviluppatori sanno, per esperienza personale, quanto sia difficile installare il server Web Apache e successivamente aggiungere e configurare il supporto per PHP, MySQL, Perl o altro. L'obiettivo degli sviluppatori di XAMPP è quello di fornire agli sviluppatori un modo semplice e veloce per installare una piattaforma di sviluppo Web completa, che ruota attorno al server Web Apache (capostipite di questa tipologia di software è stato EasyPHP). Al momento sono disponi-

bili le versioni per GNU/Linux, Solaris e MS Windows. Quest'ultima, disponibile per i sistemi operativi Windows 98, NT, 2000 e XP, installa in un colpo solo Apache, MySQL, PHP più PEAR, Perl, mod_php, mod_perl, mod_ssl, OpenSSL, phpMyAdmin, Webalizer, Mercury Mail Transport System for Win32 and NetWare Systems v3.32, JpGraph, FileZilla FTP Server, mcrypt, Turck MMCache, SQLite, e WEB-DAV + mod_auth_mysql
Directory /Xampp

Mantis 0.19

Ottima soluzione per la gestione dei Bug

Potete stare certi che qualunque sia il numero di test effettuati prima di mettere un software in commercio, nel momento in cui avrà successo e sarà utilizzato dal grande pubblico verrete sommersi da indicazioni di Bug o richieste di nuove feature. Mantis è una web application che consente agli utenti registrati di indicare un bug reperito nella vostra applicazione di modo che voi possiate avere un comodo registro dei bug, e utilizzarlo per risolverli ed elaborare i fix in modo efficiente.
Directory /Mantis

SuperEdi 3.7

Un editor piccolo e funzionale per gli sviluppatori

Ideato appositamente per gli sviluppatori, SuperEdi può essere utilizzato sia per lo sviluppo in locale che per modificare file in remoto.
Completamente gratuito, presenta tutte le principali funzionalità degli editor più blasonati: evidenziazione sintattica per i maggiori linguaggi, filtri per la manipolazione automatica del testo e supporto multilingua.
Directory /Superedi

Jasper Reports 0.6

Risolve il problema dei reports in Java

Molto probabilmente il framework più completo per generare dei reports in applicazioni Java. Ottime funzione di aggregazione e di generazione di grafici statistici. Non complicato dal punto di vista delle funzionalità, fino a qualche tempo fa mostrava ancora una qualche fragilità. La versione 0.6 che vi presentiamo è invece sufficientemente stabile e consente di generare report partendo da una base di dati e utilizzando classi Java senza avere troppi problemi
Directory /Jasperreports

Spe 0.7.3

L'editor per Python

Molti di voi avranno cominciato ad apprezzare il linguaggio Python. Spe è un editor molto evoluto che vi consente di elaborare le vostre applicazioni Python usufruendo del supporto alla syntax highliting e alla sintax completion. Si tratta di un ottimo editor che aumenta di molto la vostra produttività se state utilizzando il linguaggio Python
Directory /SpeEditor

EMF

Eclipse Modeling Framework

Si stratta di un framework di modellazione e di code generation che a partire da un modello XML fornisce classi Java in grado di operare su quel tipo di modello.
Non è decisamente un tool semplice da utilizzare ma è un requisito essenziale per installare l'Eclipse Visual Editor che vi presentiamo in questo stesso numero di ioProgrammo.
Directory /EMF

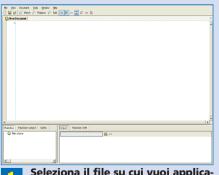
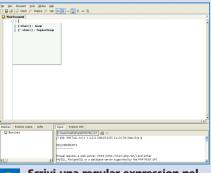
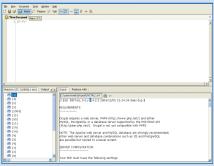
Gef

Graphical Editing Framework

Consente agli sviluppatori di creare editor grafici a partire da un modello

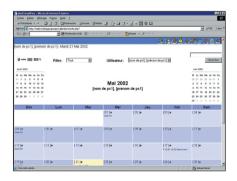
Regulator 2.0.3

Per non avere problemi con le regular expression

Le regular expression rappresentano uno dei sistemi più potenti che qualunque linguaggio porta in dotazione per i programmatori. Vengono utilizzate per effettuare ricerche su testi o stringhe SQL, o per sostituire una stringa con un'altra e per molto altro ancora. Regulator è un tool per testare e costruire espressioni regolari in modo semplificato
Directory /Regulator

1 Seleziona il file su cui vuoi applicare le regular expression utilizzando il tasto "input" posto nel riquadro in basso a sinistra. Dopo averlo aperto il file in questione comparirà all'interno del riqaudro

2 Scrivi una regular expression nel riquadro superiore, utilizzando se vuoi la code complexion per aiutarti a digiare correttamente l'espressione regolare. Da notare che all'interno del tooltip compaiono spesso degli aiuti

3 Quando avrai finito utilizza "match" oppure "replace" per testare la validità della tua regex. Utilizzando il tasto match, compariranno nel riquadro di destra tutte le occorenze dell'espressione ricercata

preesistente. Anche questo tool non è particolarmente semplice da utilizzare in applicazioni di produzione ed anche in questo caso è però un requisito essenziale per installare l'ottimo plugin di eclipse: Visual Editor
Directory /Gef

Eclipse Visual Editor
Per rendere Eclipse RAD
Eclipse è probabilmente il tool più usato per programmare in Java. Ha dalla sua una serie di facilities che aiutano il programmatore a districarsi nella complicata gerarchia di classi offerta dal linguaggio. Manca ad Eclipse, forse , forse solo la possibilità di usufruire di un tool per la generazione visuale delle interfacce. Questa possibilità viene offerta ad eclipse da uno dei suoi Plugin più amati: il Visual Editor, tramite il quale si può appunto estendere eclipse affinché abbia la possibilità di progettare visualmente le interfacce grafiche delle applicazioni.
Directory /Visual Editor

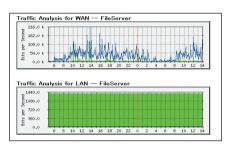
PHPGroupware 0.9.16
Per la gestione di gruppi di lavoro

La normale gestione della produzione in un ambiente di lavoro richiede ai nostri tempi uno sforzo non indifferente. La gestione dei tempi, la condivisione delle informazione, la normalizzazione dei progetti sono tutte operazioni complesse che impegnano ingegneri gestionali e manager. PHP-Groupware è un ottimo supporto alla gestione di un'attività di produzione. Si tratta di una web application da utilizzare in una intranet o anche da condividere fra uffici o rappresentanti geograficamente dislocati. Serve a coordinare l'attività di produzione a partire da una commessa fino alla consegna dei materiali, tenendo sotto controllo i costi e tutti gli altri parametri che contribuiscono alla buona riuscita della produzione.
Directory /PHPGroupware

Ndoc
Per creare facilmente codice documentato
Creare codice ben documentato è ormai una necessità. Spesso e volentieri ci troviamo a lavorare con più mani sullo stesso codice, ed è importante documentare classi e metodi affinché chiunque si trovi nelle condizioni di manipolare il codice possa farlo nella maniera migliore. Ndoc sfrutta la reflection per esaminare l'assembly generato dalle vostre applicazioni, estraendo i commenti che avete inserito e generando un file XML da sfruttare per generare la documentazione opportuna.
Directory: /Ndoc

PHTML Encoder 3.8
Per criptare le vostre pagine PHP
Avete necessità di proteggere i vostri script PHP da occhi indiscreti? Ecco a voi un enconder che vi consente di codificare i vostri script prima di distribuirli. Poiché la tecnica usata è relativa proprio a PHP, i vostri script codificati funzioneranno su ogni piattaforma, sia Windows che Linux.
Directory /PHTMLEncoder

Notepad++ 2.9
Editor per sviluppatori
Notepad++ è un editor di codice per programmatori e un potente editor di testi per utenti comuni: leggero, versatile e altamente configurabile che integra numerosi algoritmi di codifica crittografica e alcuni utili tool. Il programma è realizzato interamente in C++, mentre i linguaggi supportati sono: C/C++, Java, HTML, XML, PHP, JavaScript, Visual Basic, Perl, Python, CSS e tanti altri. Notepad++ è distribuito con licenza GNU GPL e alcune sue parti derivano dal progetto Scintilla.
Directory: /Notepad

Nant
L'assemblatore di progetti
Nant è un tool che consente di costruire un processo automatico per la generazione di applicazioni. Da utilizzare quando un prodotto è suddiviso in molte dll o sottoprodotti gestiti da reparti separati, nant crea una specie di processo batch che raccoglie i file dalle directory dei singoli sviluppatori, li compila e li predispone per la costruzione di un'applicazione completa
Directory /nant

Mrtg 2.11
Per tenere sotto controllo il consumo delle risorse

Mrtg è una Web Application scritta in Perl che consente di visualizzare graficamente il consumo delle risorse di una macchina prelevando le informazioni dal servizio SNMP. Consente ad esempio di visualizzare il consumo di banda, il numero di accessi al web server, il numero di accessi FTP, o anche l'occupazione della memoria e l'utilizzo dell'Hard Disk. Si tratta di una web application molto utile soprattutto per chi fa hosting, ma anche per chi essendo ospitato su un server vuole conoscere esattamente le prestazioni della macchina su cui il proprio sito gira.
Directory /mrtg

Nunit
Il costruttore di test
Qualche volta, anzi molto spesso vi troverete a costruire dei test per "stressare" la vostra applicazione .NET e vedere come reagisce. Nunit è un tool opensource che vi mette a disposizione un linguaggio per scrivere procedure di test. Inoltre consente di visualizzare i risultati tramite un'interfaccia grafica.
Directory /nunit

VERSIONE COMPLETA

GFI Network Server Monitor

I servizi offerti via rete sono ormai tantissimi. GFI Network Server Monitor automatizza il controllo sulla Rete, e vi avverte se qualcosa non va

GFI Network Server Monitor versione 6 è uno di quei tool da ritenersi indispensabili per la corretta gestione di un ambiente di rete. Documentare completamente tutte le funzioni di cui dispone è compito di una guida dettagliata, per quello che ci riguarda è invece importante per porre l'accento su quella che è la sua funzione primaria, ovvero eseguire dei check ripetitivi a scadenze temporali ben definite di tutti o parte dei servizi che compongono il network ed alertare con un qualunque modo, email, sms, alert un responsabile affinché possa intervenire sul servizio ed eventualmente ripristinarlo. Ma cosa si intende per servizio di rete? Si intende tutto ciò che in un qualche modo espone delle funzionalità ai partecipanti di un network.
Ad esempio è possibile controllare costantemente che il server web stia funzionando correttamente, o il dns, o il mail server e perché no il file server, o il controller del dominio. Tutti questi sono servizi essenziali. Il mancato funzionamento potrebbe provocare dei danni notevoli alla rete aziendale che servono. Per tale motivo è importante in caso di malfunzionamento che vengano automaticamente allertati tutti coloro che a qualsiasi titolo possono ripristinare il servizio.
Questa seconda importante parte, ovvero il notify può essere svolta da GFI Network Server Monitor con varie modalità. Una particolarmente interessante è quella relativa all'invio di un SMS di allerta.
L'ultimo punto su cui vogliamo porre l'accento è quello relativo all'estraneità di GFI Network Server rispetto alla piattaforma di esecuzione dei servizi. Nonostante l'applicativo giri su piattaforma Windows, può facilmente controllare infatti un server web operante su piattaforma Suse Linux. La Semplicità con cui queste operazioni possono essere svolte è sbalorditiva.

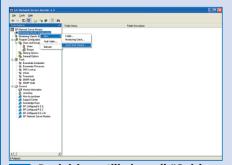
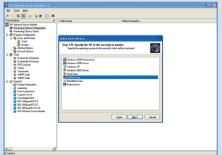
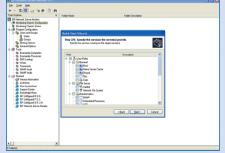
OTTENERE IL CODICE DI SBLOCCO

GFI Network Server Monitor è un tool altamente professionale. Il valore commerciale in licenza completa per tre server è di 250 €. In questo numero di ioProgrammo trovate una versione completa freeware valida per trenta giorni. Per ottenere gratuitamente la licenza per tre server dovete connettervi al sito http://www.gfi-italia.com/io e compilare il modulo di richiesta. Vi verrà inviato un codice che sblocca il prodotto per il controllo di tre server.

GFI Network Server Monitor 6

Produttore: GFI
Sito web: http://www.gfi.com
Prezzo: dai 425 € -1750 € a seconda del numero di server da monitorare

MONITORIAMO I SERVIZI

Tre semplici passi che illustrano come aggiungere un nuovo server da monitorare. Al termine di questo wizard dovrete inserire l'indirizzo IP del server nell'apposita maschera, e dare il via all'esecuzione dei controlli. Il resto lo farà il Network Server Monitor, eseguendo i check sui servizi desiderati.

1 Per iniziare utilizziamo il "Quick Start Wizard" che ci consentirà di aggiungere al nostro pool di server da monitorare una nuova macchina, con pochi colpi di mouse

2 Scegliamo la piattaforma di riferimento su cui il server gira. Tra le numerose opzioni disponibili scegliamo una "Redhat" linux su cui ipotizziamo giri un server web

2 Individuiamo i servizi da monitorare, scegliendoli dalla lunga lista che il wizard ci mette a disposizione. Possiamo scegliere ovviamente selettivamente cosa controllare

Sezioni critiche

L'effettiva realizzazione dei costrutti di base, come fork e join oppure cobegin e coend, per l'attuazione della programmazione concorrente, si scontra con reali problemi tecnici. Vediamo quali

Questa volta parleremo delle relazioni che intercorrono fra produttore e consumatore. Non abbiate timore, non stiamo per condurvi nei meandri di una discussione socio-politico-economica. Come sempre vi parleremo, invece, di un problema reale, che avviene nei sistemi operativi quando i processi condividono fra loro una qualche risorsa o sono concatenati da una relazione.

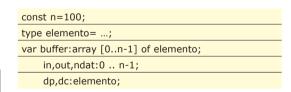
PRODUTTORE E CONSUMATORE

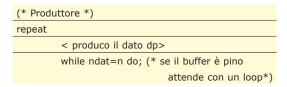
Iniziamo da un esempio che renda abbastanza semplice la comprensione del nostro problema. È abbastanza frequente che in un processo che coinvolge il Sistema Operativo esistano due attori un produttore ed un consumatore. Il primo produce risorse, il secondo le consuma. Un driver di stampa produce caratteri, una stampante consuma i caratteri, di fatto li usa per stamparli. Un compilatore produce codice oggetto che viene consumato da un assemblatore. Ancora lo stesso assemblatore produce moduli caricabili consumati da un loader.
Permettere al produttore e al consumatore un'esecuzione concorrente che aumenterebbe sensibilmente la produttività del sistema. Si tratta di accoppiare due elementi in modo che siano in grado di comunicare. La possibilità di sincronizzarli sulla cadenza di un clock non è molto efficiente, giacché il consumatore è fortemente dipendente dall'attività del produttore. Agisce soltanto quando rileva delle produzioni. Inoltre, bisogna garantire ai due uguali velocità. Un buffer potrebbe risolvere questo problema. Si tratta di un'area di memoria, dove andare a riportare i dati prodotti e prelevarli per consumarli. In questo meccanismo si rileva il limite per il produttore di agire solo quando il buffer è vuoto. Mentre per il consumatore di azionarsi solo quando il buffer è pieno. Una soluzione plausibile adotta un cosiddetto array circolare, che associato alla presenza di due indici: in e out, da usare rispettivamente per il produttore e consumatore risolve il problema precedente. Sebbene non si abbia adesso il limite di attendere che il buffer sia completamente pieno o vuoto, si riscontra comunque il vincolo dettato dalla lunghezza del buffer.

Fig. 1: **Buffer circolare per il problema del produttore e consumatore**

LE MANI NEL CODICE

Passiamo all'implementazione. Abbozziamo inizialmente una plausibile struttura dati per il buffer a struttura circolare.

```
const n=100;
type elemento= ...;
var buffer:array [0..n-1] of elemento;
    in,out,ndat:0 .. n-1;
    dp,dc:elemento;
```

La variabile ndat indica il numero di elementi che in un generico istante contiene il buffer. Inizialmente le tre variabili *in, out* e *ndat* valgono zero. Sviluppiamo adesso le due routine per produttore e consumatore. Esse non saranno altro che due cicli infiniti. Il produttore continuerà all'infinito a produrre elementi processati dal consumatore.

```
(* Produttore *)
repeat
        < produco il dato dp>
        while ndat=n do; (* se il buffer è pino
                                  attende con un loop*)
```

```
            buffer[in]:=dp;
            in:=(in+1) mod n;
            ndat:=ndat+1;
until false;

 (* Consumatore *)
repeat
            while ndat=0 do; (* se il buffer è pino
                                    attende con un loop*)
            dc:=buffer[out];
            out:=(out+1) mod n;
            ndat:=ndat-1;
            <consumo il dato dc>
until false;
```

Il comportamento delle due routine è di facile comprensione. Il produttore dopo aver prodotto il dato lo consegna al buffer solo nel caso in cui questo non sia pieno. Questa ultima operazione si realizza controllando che l'apposito contatore ndat non abbia raggiunto la capacità massima *n*. Successivamente, al fine di incrementare l'indice di riferimento per il consumatore in, si usa l'operazione di resto mod proprio per tenere conto della circolarità del buffer. Infatti, se l'indice avesse sforato di uno la capacità massima *n-1*, applicando l'operazione mod si garantisce che questi riparta da zero. Ovviamente, al termine si incrementa il contatore dei dati del buffer. Il processo del consumatore funziona in modo speculare.

C'È UN PROBLEMA, BISOGNA SUPERARLO

Sebbene le due procedure funzionino correttamente per una esecuzione separata, esse potrebbero provocare dei problemi per il caso concorrente. Vediamo come. Preannuncio che l'operazione che provoca dei problemi è proprio l'incremento della variabile *ndat*. Per comprenderne il motivo è necessario verificare l'espansione di tale istruzione in linguaggio macchina. Ecco come si implementa in assembler:

```
load r1, m(100)
incr r1
store m(100),r1
```

Analogamente per decremento:

```
load r1, m(100)
decr r1
store m(100),r1
```

Con *r1* un registro della cpu e *m(100)* l'indirizzo di memoria, per ipotesi 100, dove è contenuta la variabile *ndat*. Se il processo può essere in qualsiasi momento interrotto si possono ottenere degli effetti indesiderati. Supponiamo che per sei distinti e sequenziali istanti di tempo si eseguano le seguenti sei istruzioni macchina.

Viene riportato in tabella 1 il contenuto dei registri e delle locazioni di memoria sensibili. Supponiamo che inizialmente la variabile ndat corrispondente alla locazione *m(100)* valga 5.

t0:	produttore esegue	*load r1, m(100)*	*r1* vale 5	
t1:	produttore esegue	*incr r1*	*r1* vale 6	interruzione
t2:	consumatore esegue	*load r1, m(100)*	*r1* vale 5	
t3:	consumatore esegue	*decr r1*	*r1* vale 4	interruzione
t4:	produttore esegue	*store m(100), r1*	*ndat* vale 4	
t5:	consumatore esegue	*store m(100)*, r1	*ndat* vale 4	

Tabella 1: il flusso delle istruzioni relative al programma di esempio

Si nota come dopo aver eseguito le prime due istruzioni assembler, corrispondenti all'incremento per il produttore, un'interruzione passa il controllo al processo consumatore. Nell'istante *t2* però sebbene il processo produttore abbia incrementato il registro *r1*, questo non è stato ancora memorizzato. Così, se avviene una seconda interruzione, dopo l'istante *t3* le finali istruzioni assembler di store riprendono valori non corretti. Infatti, ci aspettammo, dopo un'operazione di produzione e una di consumo, ossia un incremento e un decremento, una valore uguale a quello iniziale. Purtroppo ndat non risulta 5 bensì 4. Il risultato è comunque sbagliato se nell'espansione assembler anziché usare un solo registro per di due processi il compilatore si servisse di due distinti registri. Si deduce che si arriva ad un risultato scorretto se si consente di manipolare concorrentemente l'istruzione di incremento o decremento. In definitiva si devono attuare delle correzioni per superare l'effetto indesiderato. Ad ogni modo è evidente la violazione delle condizioni di Bernestein.

FORMALIZZAZIONE DELLE ESIGENZE

Focalizziamo l'attenzione su un insieme di processi cooperanti *P1, P2, .. Pn*. Ognuno di essi ha un segmento di codice chiamato sezione critica o regione critica, porzione di codice condivisa. Qui ogni processo condivide variabili. Non permettendo l'accesso a tale sezione ai rimanenti processi si evitano errori del tipo sopra descritti. L'accesso alla sezione critica deve essere garantito nel tempo in modalità mutuamente esclusiva. Ovviamente, al più un processo deve essere all'interno dell'area. Il protocollo di uso di tale sezione è fatto in modo che un processo richieda l'utilizzo della sezione critica. Una volta entrato svolge le mansioni relative in modo che nessun altro processo entri. Alla fine del processo si rilascia l'area. Il codice che resta da eseguire è la parte rimanente. Ecco come per ogni processo può essere

Utilizza questo spazio per le tue annotazioni

NOTA

CONDIZIONI DI BERNESTEIN

Due istruzioni *S1* e *S2* sono eseguibili in modo concorrente se seguono le condizioni di Bernstein:

1. ***R(S1)* intersezione *W(S2) = {}***
2. ***W(S1)* intersezione *R(S2) = {}***
3. ***W(S1)* intersezione *W(S2) = {}***

***R* è l'insieme di input, *W* è l'insieme di outout.**

DAL PROGRAMMA AL PROCESSO

Fork e join e cobegin e coend producono nuovi processi. Bisogna quindi distinguere i due elementi. Un processo sequenziale in prima istanza è definito come l'esecuzione di un programma. Il programma è un'entità passiva, il processo è attiva.

formalizzata la sequenza di azioni:

```
(* generico processo che richiede, usa e libera una
                                          regione critica*)
repeat
            <sessione di entrata>
            <regione critica>
            <sessione di uscita>
            <codice rimanente>
until false
```

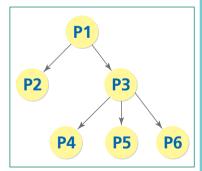
La soluzione deve soddisfare le seguenti richieste:

- **Mutua esclusione:** se un processo è nella regione critica allora nessun altro processo è in tale sezione;
- Se non ci sono processi in esecuzione nella regione critica e altri processi ne fanno esplicita richiesta, allora uno di essi deve entrare. La decisione non può spettare ai processi che si trovano nella parte rimanente. Inoltre, non si può posticipare indefinitamente nel tempo tale decisione.
- **Attesa limitata:** deve esistere un limite superiore di numeri di processi che entrino nella regione critica prima di un generico processo. Ogni qual volta si utilizza tale strumento bisogna verificare che siano soddisfatti i sopraccitati vincoli.

L'ALBERO DEI PROCESSI E LA LORO GESTIONE

È utile ricordare che un processo può generare un altro processo. Ad esempio il costrutto fork che attua la programmazione concorrente genera per il processo Pi un altro processo Pj. Il processo di creazione e dipendenza tra processi può essere espresso da un grafo dei processi. Quando un processo crea un nuovo processo si possono attuare alcune implementazioni:

- **Il padre continua a eseguire concorrentemente con il figlio;**
- **Il padre attende la terminazione del processo figlio;**
- **Il padre è il figlio condividono le stesse variabili;**
- **Il figlio condivide con il padre solo un sottoinsieme di variabili di questo ultimo.**

I primi due punti riguardano l'esecuzione mentre i secondi due la condivisione di variabili. Il sistema operativo deve attuare delle politiche, quindi scegliere tra i punti appena descritti.

Grafo dei processi. Il nodo che genera un altro processo è detto padre. Il nodo generato è il figlio

T0:	*Pi* esegue l'istruzione *while* e trova *flag[j] false*		interruzione
T1:	*Pj* esegue l'istruzione *while* e trova *flag[i] false*		
T2:	*Pj* porta *flag[j]* a *true*	entra nella sezione critica	interruzione
T3:	*Pi* porta *flag[j]* a *true*	entra nella sezione critica	

Tabella 2: a

PER TENTATIVI TROVIAMO LA GIUSTA SOLUZIONE

Sviluppiamo una serie di esempi per verificare l'effettiva garanzia della mutua esclusione per regioni critiche. Restringiamo dapprima l'attenzione ad algoritmi che sono applicabili a due soli processi: *Pi* e *Pj*. Una esecuzione concorrente che richiederà quindi la mutua esclusione è quella che deriva dall'applicazione ai due processi del costrutto *cobegin… coend*.

```
<dichiarazione delle variabili comuni>
cobegin
            Pi;
            Pj;
coend;
```

Ogni soluzione avrà la struttura proposta nel codice *(* generico processo che richiede, usa e libera una regione critica*).*

Soluzione 1 - Proviamo ad adottare una variabile turno che assegni la regione critica. Se turno vale i il processo, ottiene la sezione critica altrimenti si passa il turno al processo *j*.

```
(* Algoritmo 1*)
repeat
            (*sessione di entrata*)
            while turno <> i do;
              <regione critica>
            (*sessione di uscita*)
            turno:=j;
            <codice rimanente>
until false
```

Tale soluzione non funziona. Viene rispettata la mutua esclusione. Non si garantisce il progresso, poiché se turno vale *j* e il processo *j* sta facendo operazioni non inerenti la concorrenza, allora *Pi* non entrerà mai nella regione critica.

Soluzione 2 - Si introduce il concetto di flag. Se ad ogni processo assegniamo una bandierina il processo la alza quando entra nella sezione critica. L'altro processo verifica se la bandierina è alzata nel qual caso non entra. *flag* è un array di booleani.

```
(* Algoritmo 2*)
repeat
             while flag[j]  do;
            flag[i]:=true;
 <regione critica>
            flag[i]:=false
            <codice rimanente>
until false
```

Verifichiamo attraverso un esempio se sono verificati tutti i vincoli. Supponiamo che inizialmente i due

flag siano entrambi falsi. Inizialmente si attende che il processo *j* liberi la sezione critica. Quando alza la bandierina entriamo. La prima azione di *Pi* è alzare la propria di bandierina. Poi occupa la regione critica. Al termine si abbassa il flag. Purtroppo siamo costretti a scartare anche questa di soluzione. Non viene assicurata la mutua esclusione. Analizziamo il caso limite che lo dimostra. Consideriamo la seguente sequenza di eventi per istanti di tempo consecutivi, **Tabella 2**. Il fatto che arrivi una interruzione prima che *Pj* stia per alzare la bandierina fa sì che tutte e due i flag possano essere *true*. Si viola il vincolo sulla mutua esclusione. Entrambe i processi possono entrare nella sezione critica. L'algoritmo è dipendente dal timing dei due processi.

Soluzione 3 - Per risolvere il problema prima esposto si tenta la soluzione di alzare subito la bandierina. Ecco come si procede:

```
(* Algoritmo 3*)
repeat
            flag[i]:=true;
 while flag[j]  do;
 <regione critica>
            flag[i]:=false
           <codice rimanente>
until false
```

Abbiamo aggiustato un guasto ma ne abbiamo generato uno nuovo. La mutua esclusione è assicurata grazie al settaggio immediato del flag che avviene ancora prima che il ciclo *while* si avvii. Non è garantito il progresso poiché, se avvenisse un'interruzione subito dopo l'alzata della bandierina, si otterrebbe il seguente caso limite di malfunzionamento.

T0: Pi porta flag[i] a true	interruzione
T1: Pj porta flag[j] a true	

Si innescano per i due processi due cicli infiniti che lasciano in loop l'intero sistema.

Soluzione 4 - Si ottiene finalmente un risultato funzionante se si mescolano i metodi fino ad ora analizzati. Questa è la soluzione di Peterson, un vero esperto nell'ambito dei sistemi operativi.

```
(* Algoritmo 4*)
repeat
            flag[i]:=true;
turno:=j;
 while (flag[j] and turno=j) do;
 <regione critica>
           flag[i]:=false
           <codice rimanente>
until false
```

Per entrare nella sezione critica devono verificarsi simultaneamente due condizioni. Il flag e il turno devono essere associati a valori opportuni per il processo. Entrambe le variabili vengono assegnate fuori dal ciclo. La garanzia della mutua esclusione, anche in presenza di una sequenza sfavorevole di interruzioni, è dovuta alla variabile turno. Questa non può valere contemporaneamente due valori diversi, cosicché al più un processo occuperà la regione critica. Anche il progresso è garantito. Per tale vincolo il punto critico è il *while*. Qui possono verificarsi due situazioni. La prima è che *Pi* vuole entrare e *Pj* no, in tal caso si può notare come *Pi* entri. La seconda è che si abbia più di una richiesta di entrata. Un solo processo entrerà certamente nella sezione critica così come descritto per il caso precedente della mutua esclusione.

Soluzione 5 - La generalizzazione a n processi fa riferimento proprio alla soluzione appena trovata. Questa volta il flag potrà valere uno tra tre diversi stati: inattivo *(inat)*, pronto a entrare *(pe)*, nella regione critica *(nrc)*. Per completezza di trattazione il codice è presentato di seguito.

```
repeat
   repeat
      flag[i]:=pe;
      j:=turno;
      while i<>j do
         if flag[j]<>inat
         then j:=turno;
         else j:=(j+1)mod n;
         flag[i]:=nrc;
      j:=0;
      while (j<n) and (j=i or flag[j]<>ncr) do
         j:=j+1;
   until (j>=n) and (turno=i or flag[turno]=inat);
   <regione critica>
   j:=(turno+1) mod n;
   while (flag[j]=inat) do
      j:=(j+1) mod n;
   turno:=j;
   flag[i]:=inat;
   <parte rimanente>
until false;
```

Esistono anche altre soluzioni per la risoluzione del problema.

CONCLUSIONI

L'analisi approfondita della programmazione concorrente svela appieno le sue difficoltà di sviluppo ma anche i molteplici aspetti di interesse. Il viaggio alla scoperta di questo ambito continua tra queste pagine. Esplorando algoritmo, dopo algoritmo avremo la possibilità di avere una visione più completa sui sistemi operativi.

Fabio Grimaldi

TERMINAZIONE DI PROCESSI UNIX

Un processo termina naturalmente quando ha eseguito tutte le istruzione. Ad ogni modo esistono altri casi in cui un processo padre è costretto a "uccidere" un processo figli. In Unix si tratta di lanciare il comando kill id; dove id è l'identificativo del processo de eliminare (uccidere). Ad esempio associato al lancio di una fork. L'esigenza di eliminare un processo in modo non naturale può scaturire da due ragioni:

- **Quando il processo non svolge attività;**
- **Quando il processo eccede nell'uso di alcune risorse, provocando di fatto instabilità nel sistema.**

Esiste anche un altro caso in cui un processo viene ucciso. Quando il processo padre o comunque un processo ad un livello più alto è eliminato. Si parla in questo caso di terminazione a cascata.

ON LINE

SOURCEFORGE

Istituzionale. Il sito principe per la consultazione e la creazione di progetti OpenSource. È stato il primo a fornire la possibilità di aprire un CVS OnLine, dove per CVS si intende un sistema di upload e revisione del codice che garantisce la condivisione delle risorse.
http://www.sourceforge.net

GOTDOTNET

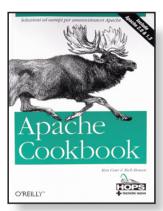
L'antagonista. La forza dirompente dell'OpenSource sta dimostrando nel tempo di essere un motore per l'evoluzione dell'informatica in generale e del segmento programmazione in generale. Cosi' è nato GotDotNet, si tratta anche in questo caso di un sistema che mette in grado i programmatori di riunirsi in gruppi e rendere disponibile il loro lavoro tramite una piattaforma ci CVS.
http://www.gotdotnet.com/

FREEVBCODE

Quello più nuovo. Si tratta di un sito appartenente al circuito DevX.com il che è già di per se garanzia di qualità. Al suo interno trovano posto centinaia di progetti e di snippet di codice relativo al Visual Basic.
http://www.freevbcode.com/

Biblioteca

APACHE COOKBOOK

Che Apache sia uno dei server Web più diffuso al mondo è ormai noto. Nonostante l'agguerrita concorrenza dei rivali, la sua diffusione è tale che con buona approssimazione si può dire che Internet e Apache formano un binomio indissolubile. Nonostante questo, soltanto una minima parte delle potenzialità di Apache vengono sfruttate dai sistemisti.
Si tratta infatti di un sistema complesso, estendibile per mezzo di moduli, e molto configurabile. Ci sono centinaia di opzioni che se ben conosciute possono risolvere in modo semplice molti dei problemi che normalmente si incontrano nel mettere online un sito Web. Questo libro di Ken Koar e Rich Bowen, ha per titolo lo "Apache CookBook", volendo tradurre letteralmente in italiano suonerebbe come qualcosa del genere "Ricette pronte per usare Apache". Si tratta insomma di un libro molto pratico, e dobbiamo dire, anche molto ben scritto, comprensibile a qualsiasi livello che mostra all'utente le diverse opzioni possibili per una buona configurazione del web server. Non possiamo dire che si tratti di un libro indispensabile, tuttavia leggendolo si ha la sensazione che molte cose che prima potevano sembrare "oscure" trovano invece una precisa collocazione.

Difficoltà: Bassa • **Ken Koar & Rich Bowen** • **Editore:** Hops • **ISBN:** 88-481-1652-3 • **Anno di pubblicazione:** 2004 **Lingua:** Italiana • **Pagine:** 238 • **Prezzo:** € 29,90

PROGRAMMARE CON IL .NET FRAMEWORK

Se fino a qualche tempo fa si poteva dire che C# era un linguaggio emergente, adesso si può dire che si tratta di un linguaggio consolidato. Alle sue spalle il .NET FrameWork se pur con tante difficoltà comincia a diventare un punto di riferimento per moltissimi programmatori. Imparare C# è una buona idea per chi vuole avvicinarsi al mondo della programmazione. Questo libro introduce ai concetti base del mondo . NET illustrando l'idea di programmazione ad oggetti, per poi addentrarsi nella costruzione di applicazioni Windows, console e ovviamente Web. Il libro inizia abbastanza dalle basi, consentendo a chi legge un immersione graduali nei concetti della programmazione C#, si passa da una parte propedeutica alla programmazione utilizzando i soli strumenti grafici, ovvero la GUI per poi continuare addentrandosi nei dettagli e mostrando gli aspetti più tecnici del .NET framework. Nel complesso un buon libro, utile a chi si avvicina o si vuole avvicinare a questo genere di programmazione. La nazionalità italiana degli autori garantisce una buona fruibilità della scrittura.

Difficoltà: Bassa • **Autore:** Stefano Del Furia, Paolo Meozzi • **Editore:** Mondadori Informatica • **ISBN:** 88-04-53606-3 • **Anno di pubblicazione:** 2004 • **Lingua:** Italiana • **Pagine:** 381 • **Prezzo:** € 14,80

LABORATORIO VISUAL BASIC.NET

Buona parte degli sforzi di Microsoft dell'ultimo anno tendono a facilitare il passaggio degli sviluppatori da Visual Basic a Visual Basic.NET. Il libro di Romano Gallifuoco va nella direzione di fornire un manuale pratico all'apprendimento delle nuove tecnologie. La praticità del libro si riscontra nell'elevato numero di esempi e nella quantità di esercizi, la volontà è quello di semplificare l'apprendimento di nozioni teoriche mostrando come queste trovino sempre un riscontro nella risoluzione di problemi pratici. Si parte delle basi, illustrando i controlli del framework .NET, si passa attraverso la programmazione ad oggetti per poi arrivare alla trattazione di temi corposi, quali lo sviluppo di Web Services e la programmazione della GDI.
Senza dubbio si tratta di un libro da non perdere, uno di quelli che conviene avere sempre a portata di mano nella propria libreria, o meglio accanto al computer mentre si sviluppa.

Difficoltà: Bassa • **Autore:**Romano Gallifuoco • **Editore:** Hops • **ISBN:** 88-503-2105-8 • **Anno di pubblicazione:** 2003 • **Lingua:** Italiana • **Pagine:** 376 • **Prezzo:** € 21,00